TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

UMER IFTIKHAR

EXTENDABLE FRAMEWORK FOR DATA COLLECTION AND ANALYSIS IN PRODUCTION SYSTEMS

Master of Science Thesis

Examiner: Professor Dr. Jose L. Martinez Lastra, Dr. Andrei Lobov
Examiner and topic approved by the Council meeting of the Faculty of Engineering Sciences on 31st May 2017

ABSTRACT

UMER IFTIKHAR:
Tampere University of technology
Master of Science Thesis, 64 pages, 4 Appendix pages
December 2017
Master's Degree Programme in Automation Engineering
Major: Factory Automation and Industrial Informatics
Examiner: Professor Dr. Jose L. Martinez Lastra, Dr. Andrei Lobov

Production systems constitute the backbone of any organization aiming to maximize its profits and cut down its cost. The data accumulated from numerous processes is critical to optimize the chain of operations within the company. However, the major hindrance for the successful operation of production systems is the way organizations manage their data. Normally, data is collected from diverge sources across the industry and therefore poses integration challenges for achieving interoperability. The poor quality of data collected from legacy systems is reported as being the major reason for the frequent failures of the modern systems.

For successful interoperability of heterogeneous systems, the production systems should not only accommodate the legacy systems, but also need to build a system flexible enough to support integration for the contemporary systems.

This research work aims to implement a flexible data collection and analysis framework, allowing the user to collect the data, clean it, and convert it in to the specified format and thereafter, perform desired analysis on it. The implemented work has been accomplished on a general level rather than examining a specified organization. The research is primarily divided in to theoretical and practical parts. The former part describes scientific literature regarding the entitled research topic and then shapes the ground for the practical implementation. The later part demonstrates the implementation of the uses cases for collecting, converting and analyzing the data.

The implementation has been performed on the legacy system hub developed as part of C2NET project. The function block approach has been extensively used while implementing the framework in the provided platform. The framework presents a unified platform capable enough to provide data collection, its transformation and analysis, thereby, solving the integration and interoperability issues faced by the organizations.

PREFACE

This dissertation is a consequence of strenuous efforts and unceasing struggle of several months. This period has provided me with an opportunity to undergo intense learning and have considerably added to my scientific knowledge and set of both, technical and personal skills. At this point, I will take the opportunity to reflect and regard the people who have assisted and encouraged me through this process of self-development.

Foremost, I feel highly indebted to Dr. Andrei Lobov, whose incessant guidance and valuable feedback had aided me to accomplish the research work. Moreover, this work has been carried out at the FAST Laboratory in the Factory Automation and Hydraulics Department, for which I am highly grateful to Professor Dr. Jose Luis and manager Anne Korhonen for allotting me the worthwhile opportunity to work at the FAST laboratory.

I would further like to extend my gratitude to my talented colleagues Wael and Borja for their undeterred support and cooperation. I am also thankful to my friends Usman, Shahbaz and Ahsan for helping me through the proof reading of the work.

Last but not the least; I will express my gratefulness to my family for their wise counsel, sympathetic ear and consistent motivation.

Thank you very much, everyone!

Umer Iftikhar

18th December 2017

Tampere, Finland

CONTENTS

1. INTRODUCTION 1
1.1 Motivation 1
1.2 Thesis Scope 1
1.3 Hypothesis 2
1.4 Objectives 2
1.5 Thesis Structure 2
2. LITERATURE REVIEW 4
2.1 Data Collection and Analysis 4
2.1.1 Importance of Precise Data Collection 5
2.1.2 Analysis of Data 6
2.1.3 Security for Data 8
2.2 Algorithms 9
2.2.1 Asymptotic Notations 12
2.2.2 Classification of Algorithms 14
2.3 Data Structures 15
2.3.1 Categorization of Abstract Data Structures 16
2.3.1.1 Linear Structures 17
2.3.1.2 Trees 18
2.3.1.3 Graphs 19
2.3.2 Big-O Complexities amongst Data Structures 20
2.4 Legacy Systems 21
2.4.1 Problems with Legacy Systems 21
2.4.2 Transformation of Legacy Systems 23
2.5 Production Systems 25
2.5.1 Automation in Production Systems 26
2.6 ISA 95 Standard 27
2.6.1 ISA 95 Standard 27
2.6.2 Hierarchy Levels of ISA 95 Model 28
2.7 Function Blocks 30
2.7.1 IEC 61499 Standard 30
2.7.2 PLANT-COCKPIT 31
3 METHODOLOGY 33
3.1 Approach 33
3.2.1 Architectural Overview 34
3.2 Tools and Techniques 36
4 IMPLMENTATION 38
4.1 Interaction of Systems 38
4.1.1 Interaction for the Mapper 38
4.1.2 Interaction for Adapter Instance 46
4.1.3 Interaction for Analyzer Adapter 47

4.2 Use Cases 48
4.2.1 Text Adapter 48
4.2.2 REST Adapter 51
5 RESULTS & DISCUSSION 54
6 CONCLUSION 56
6.1 Accomplishments 56
6.2 Challenges and Limitations 57
6.3 Future Prospects 57
REFERENCES 59
APPENDIX A – LEXER AND PARSER GRAMMAR 65
Lexer Grammar 65
Parser Grammar 66

LIST OF FIGURES

Figure 1: Forms of Data 4
Figure 2: Example of various forms of Data 5
Figure 3: Relation between data, information, knowledge and wisdom 5
Figure 4: The data science process [40] 8
Figure 5: Notion of an Algorithm [2] 10
Figure 6: Different algorithms for one problem 11
Figure 8: Analysis & Design process of an Algorithm [2] 12
Figure 9: Relationship between growth rates of various Algorithms [5] 13
Figure 10: Graphical Representation of Big O Notation [6] 14
Figure 11: Graphical Representation of Big Omega Notation [6] 14
Figure 12: Graphical Representation of Big Theta Notation [6] 15
Figure 13: Categories of Data Structures [8] 16
Figure 14: Singly Linked List [12] 17
Figure 15: Doubly Linked List [11] 18
Figure 16: Circular Linked List [11] 18
Figure 17: Graphical Representation of Tree [15] 19
Figure 18: Ways of Representing a Graph [18] 20
Figure 19 Common Data Structure Operations 21
Figure 20: Concerns of Legacy Systems 22
Figure 21: Production System [58] 25
Figure 22: Functional Hierarchy of Automation systems per ISA 95 28
Figure 23: 5-level automation pyramid [68] 29
Figure 24: Function Block Model [61] 30
Figure 25: Function Block Interconnection [62] 31
Figure 26: General System Overview 33
Figure 27: Workflow for the System 34
Figure 28: System Architecture Diagram 35
Figure 29: Illustration of Adapter Instance & Analyzer Adapter 36
Figure 30: Functioning of Parser to convert High Level Language 38
Figure 31: Graphical Representation for Modifying Grammar for new source 42
Figure 32: Abstract Tree for Configuration 44
Figure 33: Adapter Creation & Fetching of Data 46
Figure 34 Creation of Data Storage in the Analyzer Adapter 47
Figure 35 Querying of Data Storage in the Analyzer Adapter 47
Figure 36 Resource Manager displaying fetched data as HTML 49
Figure 37: Example of Full Configuration to LSH 49
Figure 38: SQL Analyzer Adapter Interface 50
Figure 39: Example of Analyzer Configuration to LSH 50
Figure 40: Example of Input Configuration to LSH 51
Figure 41: Example of Full Configuration to LSH 51

Figure 42: Creating Linked List and Saving to File System *52*
Figure 43: Dynamic Linked List Class *52*

LIST OF CODES

Code 1: Example of Mapping Higher Level Language to JSON *39*
Code 2: Defining Fragments *40*
Code 3: Usage of Fragments for facilitating lexer tokens *40*
Code 4: Options Tag *40*
Code 5: Defining Parser rules *41*
Code 6: Example of Usage of Lexer Rule *41*
Code 7: Defining Attributes for an Adapter *41*
Code 8: Holder for multiple Adapters *41*
Code 9: Parser Rule for converting text data *42*
Code 10: Tokens sequence for Excel Analyzer *43*
Code 11: Appending new Format in to the System *43*
Code 12: Generic Adapter Rules *43*
Code 13: Sample Configuration for Converting Text data *44*
Code 14: Managing Fields via Interpreter *45*

LIST OF SYMBOLS AND ABBREVIATIONS

AA	Analyzer Adapter
ADT	Abstract Data Types
AI	Adapter Instance
ANTLR	ANother Tool for Language Recognition
C2NET	Cloud Collaborative Manufacturing Networks
DCF	Data Collection Framework
ESB	Enterprise Service Bus
ERP	Enterprise Resource Planning
FB	Function Block
FBM	Function Block Manager
FTP	File Transfer Protocol
HDFS	Hadoop Distributed File System
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
ISA	International Society of Automation
JSON	JavaScript Object Notation
LSH	Legacy System Hub
MES	Manufacturing Execution Systems
OSGi	Open Service Gateway
PCP	PlantCockpit
PDT	Primitive Data Types
REST	Representational State Transfer
RM	Resource Manager
SFTP	SSH File Transfer Protocol
SOA	Service Oriented Architecture
SSH	Secure Shell
SSL	Secure Sockets Layer
SQL	Structured Query Language
URL	Uniform Resource Locator

1. INTRODUCTION

This section presents the basic reasons and objectives of this thesis. It will emphasize on the scope along with the motivation for the thesis. Thereafter, it clarifies the fundamental problem that this thesis would propose to solve. This section additionally mentions the limitation faced during the implementation of the presented research problem.

1.1 Motivation

Data management and analysis have ceaselessly presented countless benefits to numerous associations, however enjoying such perks cannot be achieved without overcoming the obstacles posed by the convoluted and diverge data sources. Organization have continuously been battling to find the most appropriate approaches to capture information about their interested entities. The current technological advances in the fields of networking, cloud computing and hardware, has not just transformed the way data can be managed but has also prompted huge cost reduction in achieving the respective objective. Due to the rapid advancement in the field of digital systems, production systems have increasingly become reliable on the services provided by the software's. This shift towards, the new technologies and services assists organizations to optimize their plans and let them reduce their operational costs. This reliance on the provided services is not achievable without achievement of a rigid data collection and conversion framework. Accessing data from multitude of sources for extracting, aggregation, converting and analyzing purposes remains an enormous challenge. Modifying the systems either to acquire data from the legacy systems or to tackle new data formats often requires architectural changes, thus making the system more complicated. Therefore, construction of a loosely coupled system, where organizations could simply update or manage handling of diverge data sources (without caring about the architecture), would bring much need flexibility for integration purposes. The research presented in this proposition presents a solution capable of dealing with data management and analysis issues.

1.2 Thesis Scope

This proposed work contains the exploration of gathering raw data from heterogeneous sources, cleaning & processing it, and converting it in to the desired format and consequently enabling the client to analyze the respective source. On top of it all, the proposed solution allow the user to write their own grammar to verify and interpret the correct

configuration from the clients. Therefore, this solution is primarily meant to accommodate the unforeseen data formats, in the ever-growing area of production systems. This solution not only discusses the handling of new data formats, but also allow the legacy systems to be managed in a more efficient and cost effective way. As part of the thesis, the major focus area would be the redesigning of the data collection and conversion framework, facilitating the user to acquire data from diverge sources and shape it in the desired format.

1.3 Hypothesis

In order to cater the data management and analysis issue in production systems, a data acquisition and analysis translator is required, allowing the users to securely manage unforeseen data formats by collecting them from different data sources through multiple custom built data adapters and thereafter providing mapping between various data formats.

1.4 Objectives

This main objective of this thesis is the creation of a framework allowing users an easier collection of unforeseen data formats and its conversion in the required format. Moreover, the framework exposes the converted data, so it could be processed to answer the research questions. In order to fulfill these objectives this thesis will emphasize on:

1) Acquisition of data from the specified data source.
2) Converting data to specified format.
3) Analyzing the converted data.
4) Designing a Translator, where the user can specify grammar for:
5) Fetching data from sources.
6) Mapping the data source to the desired format.
7) Analyses of the required source.

1.5 Thesis Structure

Chapter ones presents an introduction to the reader by providing the motivation, scope and the objectives of the thesis. Aside from the introduction, the thesis has five core chapters; the background studies, methodology and approach section, the implementation part, results and then the conclusion.

In the second chapter, the background studies and state of the art has been discussed. It enlightens the reader about the theories and ideas involved in the respective field. It focuses on the core issues with reference to the related concepts and provides a brief overview of the research done in the field.

The third chapter discusses the methodology and the approach embraced in order to accomplish the desired goals. It tackles the challenge of finding the right solution for the given research problem. This section presents an architectural over view to solve the given problem.

Chapter 4 extends the architectural view presented in the third chapter. It elaborates the implemented solution by providing sequence diagrams of component interactions in the system. In addition to this, the chapter also shows use cases for the implemented solution.

After successful testing of the framework, Chapter 5 presents the results and discusses them thoroughly. It provides a detailed explanation about the problems that the framework has successfully solved.

The final chapter wraps up the research work by providing accomplishments of the research, the challenges and limitations that it faced, and the future prospects that the current research has opened.

2. LITERATURE REVIEW

The major aim of this section is to set up the significance of the general field of study and recognize a place where additional contributions could be made. This chapter will provide a comprehensive structure of the exploration done so far on the Data Collection, Data Structure & Algorithms, Legacy Systems, Production Systems and Function Block and PlantCockPit. This part will facilitate the pursuer to grasp the essence of the establishing blocks of this thesis that are essential to answer the research questions.

2.1 Data Collection and Analysis

Data is a set of raw values, numbers, characters, images and sounds that are produced by abstracting world into categories, measures and other representational forms. Data serve as stepping stone for deriving knowledge and information [26]. Data is often available in unorganized and raw format, which requires processing to make it organized and extract facts. Data can take multiple forms, and therefore can be categorized as either structured, semi-structured or unstructured (as shown in Figure below) as well as quantitative or qualitative [26].

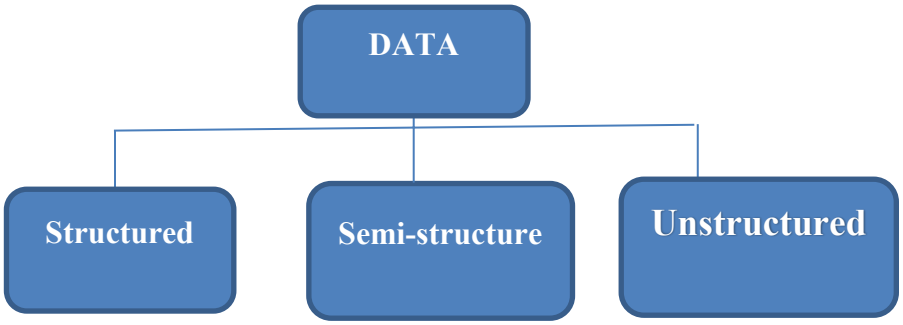


Figure 1: Forms of Data

There are two widely recognized types of the data, one is the qualitative data and other is the quantitative data. Qualitative data is non-numeric data and is collected in the form of sound, pictures or texts. Whereas, quantitative data represent numeric records expressing some properties of related objects.

Moreover, some researchers divides data based on the format and structure of the data. There are three major classifications on that basis, which are the following.

i) Structured data
ii) Semi-structured data
iii) Unstructured data

RDMS, Excel Files and SAP can be considered as structured data, whereas XML data or JSON data come under the hood of semi-structured data. Other sorts of data, e.g: audio, image or text files comes under the category of unstructured data. A small example of the above-mentioned three categories of data can be visualized is illustrated in the following Figure 2.

Figure 2: Example of various forms of Data

2.1.1 Importance of Precise Data Collection

It is quite significant to differentiate between knowledge, information and data. According to Redman, the primary source of knowledge is information, whereas information is itself derived from the data [27]. Following Figure 3 depicts the relationship between knowledge, wisdom, information and data.

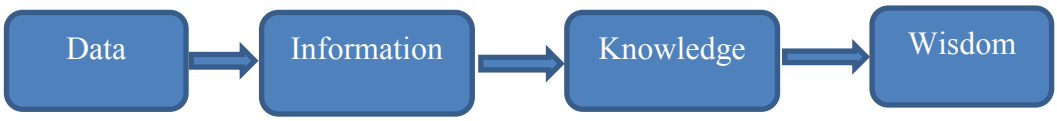


Figure 3: Relation between data, information, knowledge and wisdom

In [28], Tayi and Ballou assert that the data constitutes the raw material for the Information. However, one of the properties of data is that it can be reused for numerous purposes, unlike the physical raw materials, which are consumed only once.

"Data are intended to represent facts and without proper preservation of the context of collection and interpretation, may become meaningless" [29].

Data is the pivotal part of any sort of research studies. Although, the research conducted in one field may vary significantly than the other fields, but each of them is primarily based on some sort of data, which is then examined and thereafter used to extract information. Moreover, in today's world of ever-growing data, data has become the basic unit in the field of statistical analysis.

This ever-growing interest in the data has given rise to the importance of how data is collected and managed. In [30], Data collection is described as the process of organizing

and accumulating data. The data collection process is a vital aspect for numerous research areas. Data collection helps researchers answer their research questions. Since, the research results are directly dependent on the accumulated data; any inaccuracy in the collection of data can lead to erroneous results and can drastically affect the research interests.

However, as much as data and data collection mechanisms are important, unarguably the quality of data is also the most vital part of acceptable data. Therefore, quality of the data should be preserved during the process of data collection. However, even the quality of data is a relative concept [32], as it depends on how successfully the collected data serves the purposes of the user. Therefore, in a broader context, data quality can be referred to as "fitness for use", i.e. how convenient is it for the end use [28][33]. Whereas, Ballou and Pazer categorize the quality of data into four dimensions [34]:

i) consistent
ii) completeness
iii) timeliness
iv) accuracy

The data quality is of huge importance when data is analyzed in totality rather than individuality. In [31], Haug et al. assert that data can be merged, shared and copied in various ways, Therefore, posing special challenges to be taken care of in regards to the quality of data. Redman has summarized these hazards as follows [27]:

i) Inaccurate data
ii) Erroneous understanding of data
iii) Security and privacy of data
iv) Ambiguous data within organization
v) Poor definition of data
vi) Inconsistency among data from numerous sources
vii) Struggle in finding significant data

The importance of data collection mechanisms and the desire for high quality data can be observed by the facts and figures quoted by in [35]. According to Feldman and Sherman in [35], organizations spends nearly 30 percent of their valuable time in order to accumulate the required data. They argue that how collecting the critical data timely can differentiate winners from losers in the era of information economy. Therefore, it is evident that the precise and high quality data must be collected in timely fashion can lead to increased sales and improved productivity for the organization.

2.1.2 Analysis of Data

"Data analysis is the process of developing answers to questions through the examination and interpretation of data." [36].

In data science, the most important step after collecting the raw data is the analysis of the specified and related data to derive useful information from it. The analysis of the data has man roles to play; some of them are the following.

i) help summarize the data
ii) describe relationship among data variables
iii) identify the future outcomes

Marshall and Rosman defines the process of data analysis as the procedure to structuring and ordering of the data, as well as brining meaning to the massive data collected [37].

Moreover, Hitchcock and Hughes have pushed this definition to its limit by defining the process as ***"the ways in which the researcher moves from a description of what is the case to an explanation of why what is the case is the case".*** [38]

In the essence of signal processing data analysis has huge importance in filtering out the interference in the signals. Data analysis helps in providing a way to distinguish the signal from the noise and drawing useful inferences from the data, where the signal is the event of interest and noise can be considered as the statistical fluctuations [39].

According to Schwandt, Data analysis is an ambiguous, time consuming and a messy process, but at the same time, it is a fascinating and a creative process. The process involves making sense of the available data, interpreting it and thereafter theorizing it to retrieve broad statements among the groups of data [41]. However, researchers involved in analysis must be cognizant of the validity and the reliability of the data as well. Gottschalk [42] recognizes the following three factors affecting the reliability of analyzed data:

i) The data must be stable enough, letting the programmers to re-code the data repeatedly
ii) The data should be reproducible , allowing the programmers to classify in the same way
iii) Data must be accurate

Therefore, the data integrity is compromisable if the researchers are not capable of demonstrating the accuracy, reproducibility and stability of data analysis. In addition to this, Shamoo & Resnik [39] describe, while providing honest and accurate analysis, the statistical error likelihood must be low. This sort of challenge comprises of taking care of the following:

- Outliers must be excluded
- Missing data must be filled
- Alteration of data
- Graphical Representations of data

In [40], a complete framework for data science process is presented, starting from the collection of raw data until converting it to wisdom and something meaningful. According to Rachel Schutt and Cathy O'Neil [40], there is a lot of abundant data across the globe, that is very raw in its nature, so the first step in data science process is to collect this raw data. Once the raw data is gathered, the next step is to process this raw data to clean datasets that contain data in structured format. Python, R language and SQL are majorly used to generate clean datasets. Furthermore, Exploratory data analysis is done to fill the missing gaps that occurred during cleaning raw data. This data is then send for implementing different algorithms and models on it. According to Rachel Schutt and Cathy O'Neil [40], most people involved in the process of data science find themselves fitting in to the framework shown below in Figure 4:

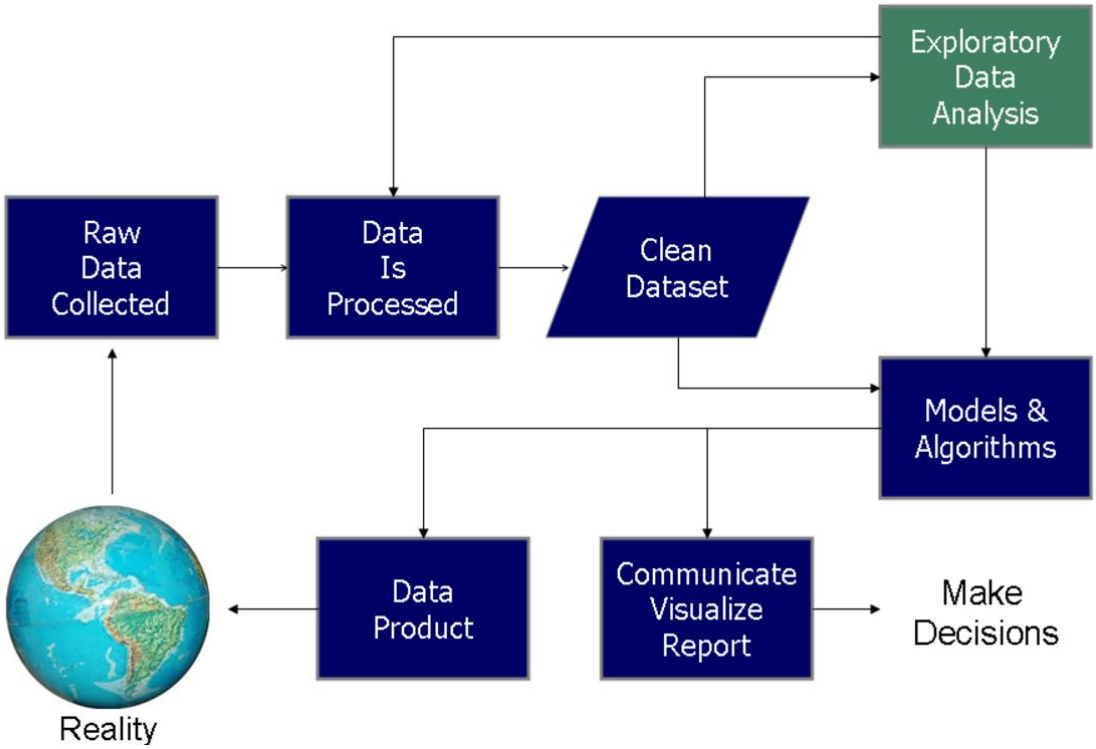


Figure 4: The data science process [40]

It is evident from the above discussion how vital it is to accurately perform data analysis in the respective research fields. As inappropriate analyses could mislead the scientific findings, resulting in misleading the readers and therefore portraying negative influence of research to the public [43].

2.1.3 Security for Data

Information security is one of the major issues worth clarification. Threats faced by the digital world mimic the threats of the physical world, e.g. If the banks can be robbed physically, the digital ones are not safe either. Vandalism, theft, exploitation, con and fraud are equally probable in the digital world as well; however, the means to achieve this

could differ. As lock picks are used in the physical world, similarly hackers use manipulation of databases and connections in the digital world [44].

Dos attacks, password-based attacks, man-in-the-middle attacks (MITM), data modification, eavesdropping, compromised-key attacks and application layer attacks are most of the commonly used attacks [45]. However, the most dangerous of the attacks are man-in-the-middle (MITM) attack, eavesdropping and data modification, since it puts the confidentially of the private/important data at risk [46]. These threats can be expected from various entities in numerous ways. Hypponen classifies hackers in to three main classes [47]:

i) Online criminals
ii) Activists groups
iii) Governments

The sole purpose of activist attacks are either ideology or to protest against a specific policy of any government or institution, whereas, the government's attacks are primarily meant for solving the crimes. To monitor underground gangs and to gather valuable evidence against them government's intelligence agencies usually use different type of hacking techniques. However, the most obvious ones are the online criminals, who operate to steal sensitive information and money for their personal use and agenda.

As discussed above there are numerous ways to bypass and sneak into the existing systems through different hacking techniques, but at the same time, there are various ways to defend against these hacking methodologies. The systems have evolved over time to incorporate the security flaws.

At the bottom of every security system, cryptology pays an important role for data encryption, integrity, identification and authenticity. To get familiar with the internet security, one should understand the certificates and cryptography. Cryptography has been there for decades, to defend messages from the unauthorized people. Navajo Indian language was used to encrypt sensitive military messages in World War 11 by the USA military [48]. Cryptography helps in modifying the original data/message so that the third party cannot apprehend the real content of the message. Confusion and diffusion algorithms are widely used to implement cryptography. In confusion algorithm, the letters of the message are replaced by the symbols, whereas in diffusion the words of the original message are shifted based on the specified algorithm. However, modern day cryptography techniques merge both algorithms to encrypt the messages [49].

2.2 Algorithms

An algorithm is a procedure for problem solving techniques which a machine or computer uses to accomplish certain goals. It is a set of elementary instructions carried in a specific

sequence to solve a specific set of problem. Generally, such a procedure undergoes three steps i.e. taking input values from the user or in any other form and then apply a set of processes and techniques on them to generate a specific set of outputs. According to [1] a specified set of commands used to accomplish certain answers is called an Algorithm. Figure 5 below illustrates the above-mentioned three steps of the algorithm procedure.

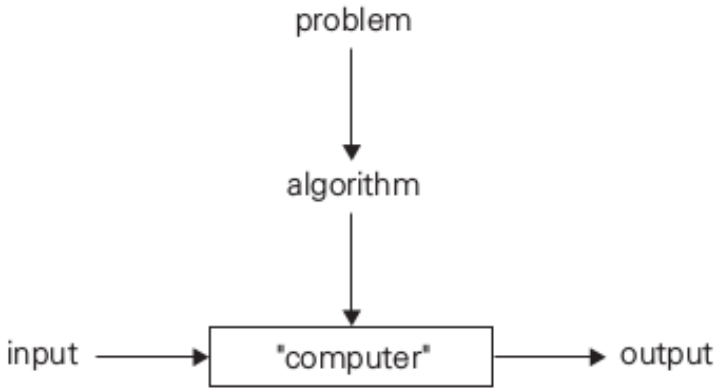


Figure 5: Notion of an Algorithm [2]

To design an algorithm a specific set of steps or cycle of steps are followed. Designing an efficient and optimized algorithm is the objective of every designer. The composition and analyzing of algorithms encompasses number of phases. The first and foremost step is to understand the problem domain. Then by choosing or merging various design techniques one designs an algorithm for the specific problem. Researchers have developed different algorithms designing techniques, some of them are discussed in [3]. According to [3] these following are the major design techniques used for creating algorithms:

1 Brute Force.
2 Greedy Algorithm.
3 Divide and Conquer.
4 Transform and Conquer.

After designing an algorithm, the next step is to communicate or express it in the form that is understandable to general audience. Pseudocode is one way to express the algorithms, it is the combination of both the natural language and programming language. There can be multiple algorithms for a single problem, and each algorithm can be implemented in different ways as depicted in the following Figure 6.

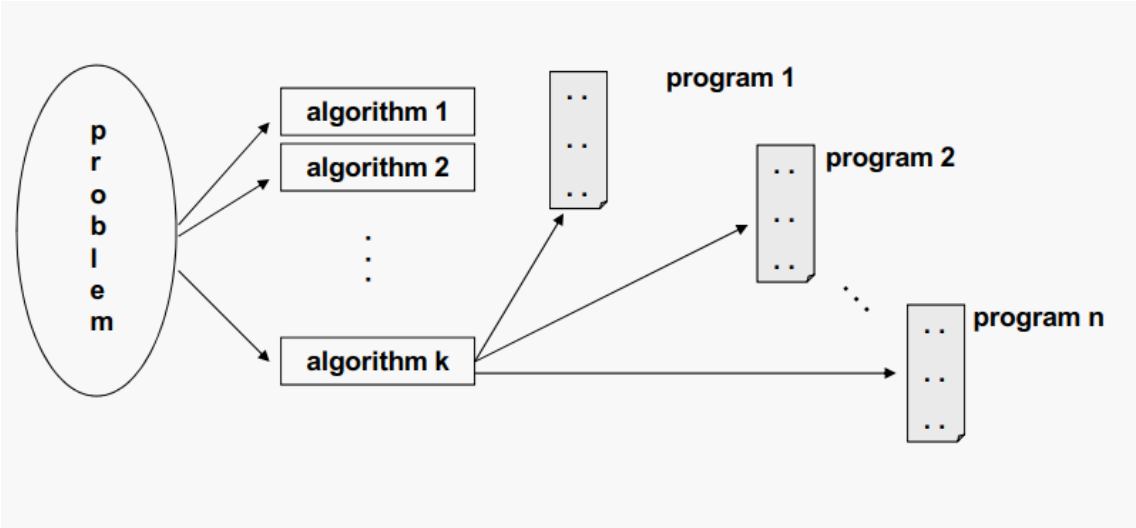


Figure 6: Different algorithms for one problem

Thereafter the algorithm must go through some sort of correctness filter or cycle to assure a correct results or outputs for authentic inputs. To test the accuracy of the algorithms various techniques are used, most common of them is induction method to prove the correctness of the algorithms. In induction method, an algorithm is tested on predefined set of inputs for which the outputs are already known. On the contrary, one only needs one instance of input yielding incorrect result, thereby resulting in the failure of the algorithm.

As we discussed previously that, it is not just designing an algorithm but to design it efficiently is the need of the hour. Preceding the correctness of the algorithm the most vital step is to check the efficiency of the algorithm. The algorithm must meet the standard efficiency criteria and must be optimal enough to solve the specified problems. Once certain efficiency standards have been met, the algorithm must be transformed in to a computer program. Designing an algorithm is an iterative process and there must be various iterations aimed at refining and improving the algorithm. Figure 6 presents the steps involved in designing a general algorithm for more or less any sort of problem.

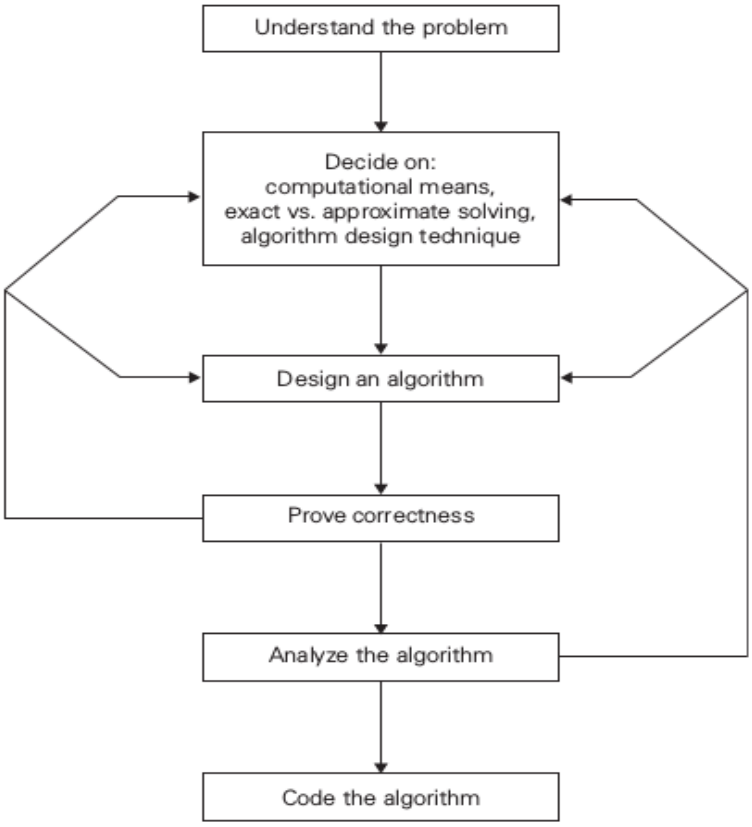


Figure 7: Analysis & Design process of an Algorithm [2]

2.2.1 Asymptotic Notations

As discussed in the previous section the next vital step after designing an algorithm is to test its efficiency. For an algorithm to be effective and reusable, it must be efficient enough. In today's industry, different scales and parameters are used to check the efficiency of an algorithm. In [4], Sedgewick et al. discusses different parameters that helps in determining and comparing the efficiency of different algorithms. These parameters provide estimate of resources required by the algorithm for solving a given problem.

Parameters discussed in [4] by Sedgewick et al. are the following:

i) Running time.
ii) Memory.
iii) Developer's efforts.
iv) Code Complexity.

Among these parameters, the most important one is the running time analysis of the algorithm. It determines the connection between the input size and the processing time required to process that input. Running time is more specifically used in analyzing how the processing time increases as the input size grows.

As mentioned in [5] the ideal way to compare algorithms is to observe their rate of growth with the growing input size, since such a computation does not depend on the execution time (which can vary from one operating system to another) or number of statements being executed (which can vary depending on programmer or specific programming language).

"**The rate at which the running time increases as a function of input is called rate of growth"**.[5]

The following diagram demonstrates the association amongst various rates of growths in order of their magnitudes:

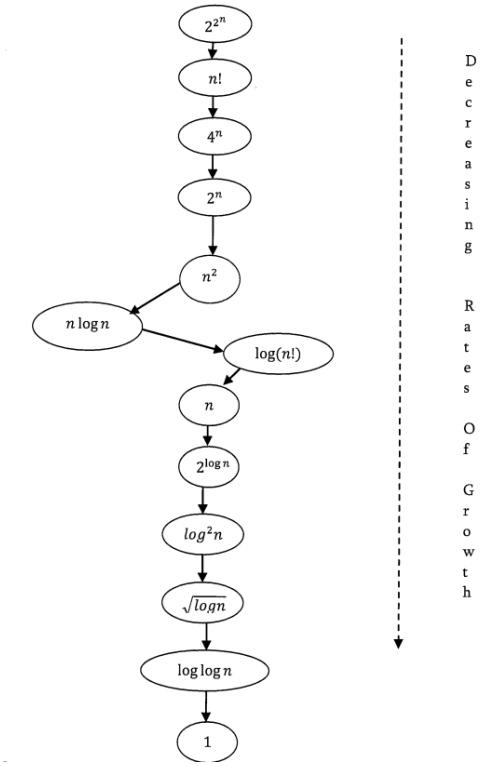


Figure 8: Relationship between growth rates of various Algorithms [5]

In [2], Levitin presents another Analysis Framework to compare the efficiency of different algorithms. The Analysis Framework in [2] is built on the following parameters.

i) Time and space efficiencies must be calculated based on the size of input.
ii) Space efficiency depends on the amount of spare memory blocks consumed.
iii) Time efficiency depends how frequently the basic operations are executed.
iv) The framework's concentrates on growth of the algorithm's running time as the size of input reaches infinity.

2.2.2 Classification of Algorithms

Asymptotic analysis allows the comparison of algorithms irrespective of any specific programming language or hardware so that we can decisively say that an algorithm is more efficient than the others are. It is too hard to find a data structure that presents ideal performance in every case. Therefore, one needs to bargain a suitable data structure for a specific task. Thereafter, one must be able to calculate how performance of every solution varies.

The analysis framework discussed earlier focuses on the algorithms order of growth. For the sake of comparison, computer scientists have tossed up three notations.

i) (big O) **O**.
ii) (big omega) **Ω**.
iii) (big theta) **Θ**.

Big O representation describes the restraining behavior of a function when the size of the input reaches approximately to infinity. For an algorithm, the Big O imposes an upper bound on its running time, thus highlighting the maximum amount of time for the completion of algorithm. Therefore, representing the "worst case scenario" of an algorithm.

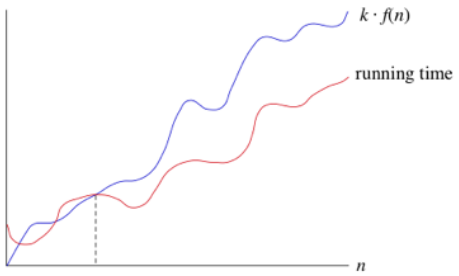


Figure 9: Graphical Representation of Big O Notation [6]

Big Ω representation signifies the minimum amount of time the algorithm will take to execute. For an algorithm, ***Omega*** imposes a lower bound on its running time, therefore, representing the "best case scenario" of an algorithm.

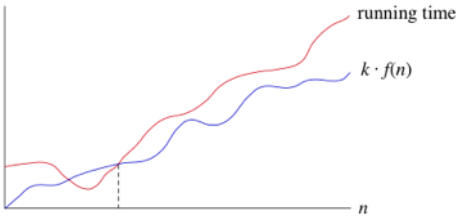


Figure 10: Graphical Representation of Big Omega Notation [6]

Big Θ representation describes that the algorithm is *Big Ω* as well as *Big O*. For an algorithm, the Big Θ imposes an tight bound on its running time. Since, the algorithms running time has been sandwiched between constants factors, therefore we call it tightly bound.

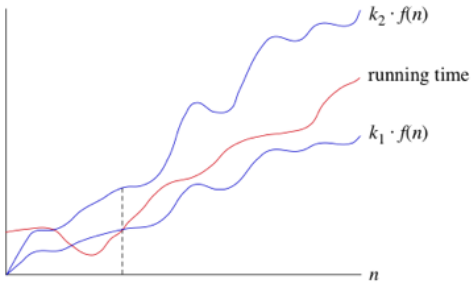


Figure 11: Graphical Representation of Big Theta Notation [6]

All of the above notations can be summarized as follow:

If an algorithm is represented as function f(n), then;

- f(n) ∈ O means that the growth of the algorithm will not surpass the upper bound.
- f(n) ∈ Ω signifies that the algorithms rate of growth will grows no slower than the lower bound
- f(n) ∈ Θ mean that the algorithm grows like the function itself.

2.3 Data Structures

Data Structures are understood as a computers format for handling large amount of incoming data. The type of data structure required for storing the desired data depends on the format of the data. Normally, before storage of the data in the specified data structure requires some sort of alteration to the data, to make it fit the format of the respective data structure. Every data structure has a reserved type, called ***'data type'***, which is used to differentiate it from other types of structures. The data type determines which sort of data can be stored in the respective structure and what operations can be performed on it. According to [7] data structures are generally divided into two types.

i) Primitive Data Types
ii) Abstract Data Types

Primitive types are available on most computers as part of their in-built features. They contain the characters, numbers, and truth-values. Normally these types are denoted as: *INTEGER, CHAR, BOOLEAN*. These primitive types are the basic building block for the construction of the more complex data structures.

Abstract data types, on the other hand are more complex data structures with the aim of solving problems that are more complex. In [9] Weiss, describes the abstract data type as:

"set of objects together with a set of operations". These are built by combining the primitive or built-in structures and are implementation independent. Moreover, the desired operations can also be associated with these structures to fulfill the required goals. The following diagram visualizes some of the most prevalent data structures:

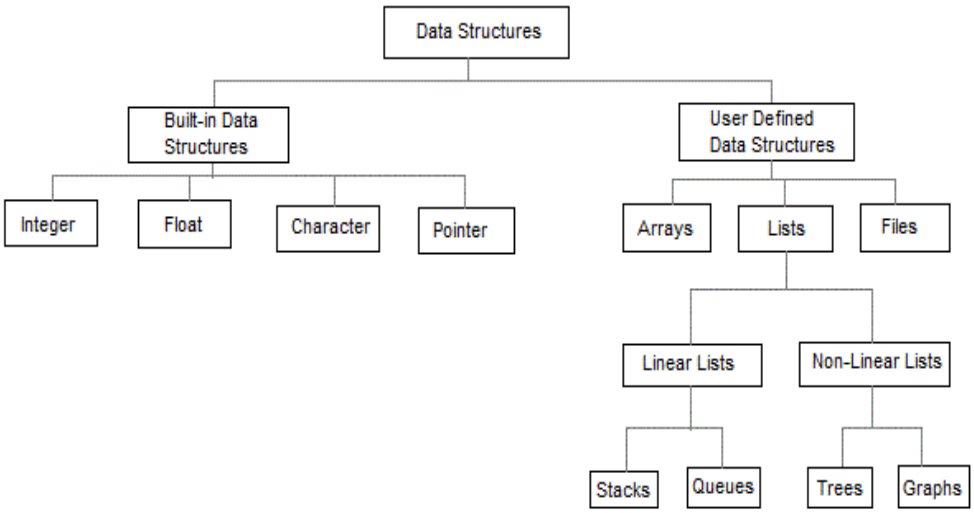


Figure 12: Categories of Data Structures [8]

The left branch of the diagram shows the primitive types, whereas the other half visualizes the abstract data types. Moreover, the diagram show, how we can leverage the already built abstract data structures and extend them to make the newer ones.

The upcoming sections will discuss the some of the most popular Abstract Data structures in detail.

2.3.1 Categorization of Abstract Data Structures

As explained earlier, the abstract data structures have the power of being shaped by the already available primitive types or the abstract types as well. Similarly, the user-defined operations could also be associated with the respective data structures. Some of the widely used operations implemented on these data structures are traversing, sorting, merging and insertion. Largely, the Abstract Data Structures can generally be divided in to three categories [10].

i) Linear Structures
ii) Graphs
iii) Trees

Each of these types are further classified in to multiple categories as well. Each of the three main categories are explained in the upcoming sections.

2.3.1.1 Linear Structures

The linear structures are the one of the simplest type of the abstract structures. The data items in the linear data structure are organized in a linear sequence, such that one item appears after another. Some of the most widely used linear structures are as follow:

i) Arrays
ii) Linked Lists
iii) Stack
iv) Queue

Although, each of these structures store data in the sequential format, however, their respective way of storing the data varies significantly. Moreover, each of them have specialized capabilities, which make them fit for the desired conditions. The main difference between the arrays and linked list is the storage of dynamic data. Whereas Stack and Queues are more specialized forms of Linked List, which are intended for storage and retrieval of data in a specific manner. The following section explains these concepts in more detail and thereafter make a comparison between arrays and the lists.

Linked list is a linear structure that stores data in form of nodes [13], where each node consists of the relevant data and location of the next element in the memory. A simple depiction of a linked list can be seen in the following figure:

Figure 13: Singly Linked List [12]

Linked list is further classified in to three main types.

i) Singly Linked List
ii) Doubly Linked List
iii) Circular Linked List

Singly linked list is displayed in Figure 13, where one node contains address for the next node in the list. The last element in the list has its address set to null, indicating end of the list. Doubly linked list on the other hand contains address for the next as well as the previous node. Therefore, giving it the freedom to traverse the list in either direction. More precise and graphical depiction of the doubly linked list can be seen below:

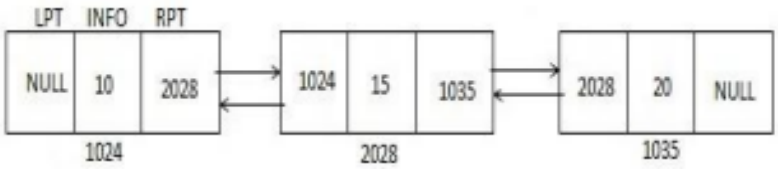


Figure 14: Doubly Linked List [11]

Circular Linked list yet add more functionality to the linked list. As shown in the Figure 15, instead of setting the link to null for the last item, it is set to contain the first item of the list, i.e: head.

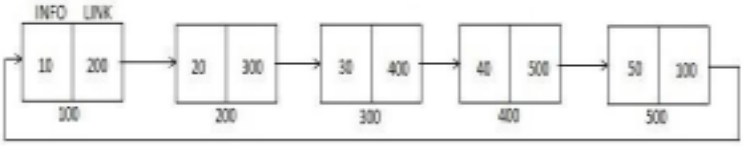


Figure 15: Circular Linked List [11]

Using linked list has several advantages over their counterparts, such as arrays. Linked lists allows to dynamically inserting the data at run time, therefore, also being widely known as dynamic data structures. The biggest advantage of the lists is that the data is not stored in continuous memory chunks, which theoretically imposes no restriction on their respective size. Moreover, this allows the insertion and deletion more efficient, as the nodes can be inserted and deleted without any performance headache (since, one only needs to change the link on specific node). However, besides all the benefits, Linked lists inherit few disadvantages as well, as part of their functionality. The first major one is the memory consumption issue, since we are storing an extra of information in the form of link (containing the address of the next node). Moreover, to access any specific node, one has to traverse the list to reach a specific location. Arrays on the other hand provide the benefit of directly accessing the data at nth index directly.

2.3.1.2 Trees

Trees belong to the hierarchical data structures category. Although formed in the same way as the linked list but differs a lot in functionality. Trees are the building blocks for numerous areas in the field of computer sciences. Database systems, graphics, operating systems and networking, all of these have trees at their core implementation.

Tree data structure share numerous attributes with their botanical cousins [14], e.g.

i) Root
ii) Leaves
iii) Branches

Root sits at the top of the tree, providing one with the entrance point for the tree. A tree consists of just one root and there could be only one path from the root connecting any

node in the tree. Leaf is the node, which has no further nodes/children. A branch is path/connection from root to any specific node in the tree. Trees are further classified in to several types. Some of the most frequently used tree data structures are:

i) Binary Tree
ii) Binary Search Tree
iii) Red-Black Tree
iv) AVL Tree
v) Heap Structure

A graphical representation of a tree can be seen as follow:

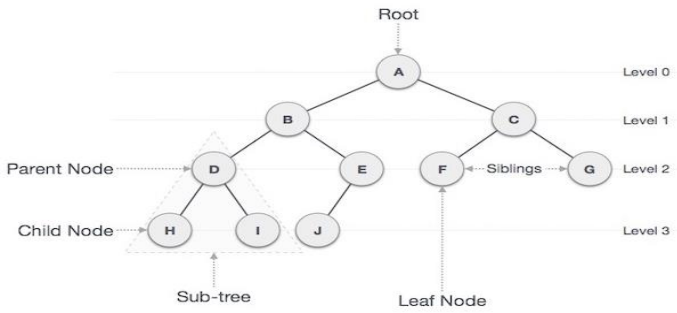


Figure 16: Graphical Representation of Tree [15]

Although, the same concept applies to every type of tree data structure, but the way data is stored, mapped and traversed varies significantly. The review discusses the most popular form of tree (*Binary Search Tree*), which is used for faster inserting and traversing purposes, significantly reducing the time for each operation.

Binary Search Trees **(BST)** are used for rapid access and storage of information [17]. Each node can have maximum of two-child nodes. It possess the following properties:

i) A nodes left sub-tree holds a key less than or equal to the key of its parent node.
ii) A nodes right sub-tree has a key greater than to the key of its parent node.

Although, the BST is recognized for its fast searching of items based on a given key [16], but in the worst-case scenario, the BST can converge to a singly linked list. However, in order to avoid such a case, ***AVL*** trees, yet another implementation of tree data structure, are implemented to maintain the balance of the BST [16]. Therefore, depending on the desired need, trees can used to store information that is hierarchical in nature, giving them the control to traverse and process the data in an efficient way.

2.3.1.3 Graphs

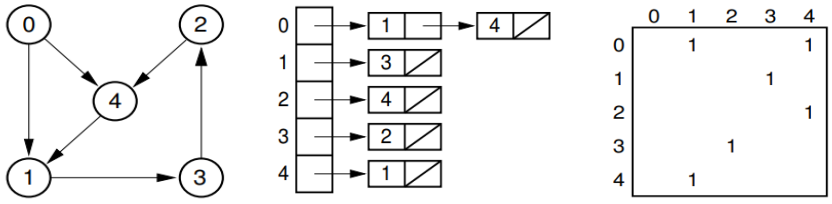
Graph data structure is yet another powerful abstraction of the abstract data structures. A graph consists of multiple sets of vertices and edges. Each vertices represents a node of the graph, whereas an edge represents a link between two vertices [19]. Moreover, a graph would most likely associate a value with each edge, commonly known as the weight. The

weight property could vary significantly depending on the type of implementation. It could be a distance, cost, length, etc. Graphs are generally divided in to two kinds.

i) Directed
ii) Un-Directed

When edges of a graph are not directed, it is called undirected graph. Whereas, graph with directed edges is categorized as directed graph (*Figure 12 a*). In an undirected graph, the connection from point A to point B and from B to A, signifies the same. On the other hand, in the directed graphs, connection from A to B and from B to A, can vary and have no relevant relationship. The two most common ways of representing graphs are [18]:

i) Adjacency Matrix (*Figure 12 c*)
ii) Adjacency List (*Figure 12 b*)



a ***b*** ***c***

Figure 17: Ways of Representing a Graph [18]

Figure 17 shows a directed graph with its adjacency list and matrix.

Following are the problem ranges where graph theories are currently being applied [18].

i) Connectivity modelling in the networking domain
ii) Dijkstran algorithm for shortest path calculation
iii) Artificial Intelligence
iv) Displaying transition from one state to the other in algorithms modelling

This section walked through numerous types of data structures, highlighting the similarities and differences each of them have. An in details analysis was presented on how to choose between various data structures depending on the required needs and the scope of the problem. Choosing the right form of data structure for the required needs can have significant impact on the performance of the process.

2.3.2 Big-O Complexities amongst Data Structures

Building an algorithm that discovers information rapidly could be the defining factor for an organizations success or failure. e.g. Googles major accomplishment originates from the algorithms allowing individuals to seek tremendous volume of data with extraordinary efficiency. There are numerous approaches to inquire and sort data. As discussed in the

previous section, scientists use asymptotic analysis to compare efficiency of the respective algorithms paying little attention to the memory or computational power of the computing device. Asymptotic analysis therefore describes the how the efficiency of the respective algorithm depends on the size of the input. Specific data structures are more suitable for one type of operation than the other, therefore, scientist typically use Big O Notations to select the data structures according to the needs of the problem. The following figure compares the efficiency of numerous operations on various data structures.[63]

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Figure 18 Common Data Structure Operations

As discussed in previous sections, Arrays are much better fit for a problem where the size of the dataset is known in advance, therefore significantly reducing the access time to each item, as compared to the Linked List. However, the use of Arrays fall apart as soon as the size of the data set becomes dynamic. Thus, Linked List or dynamic structures come in to play, but with an efficiency cost, since searching of the individual items becomes more costly as the size increases. However, diverge data structures are then looked upon to increase efficiency of the desired process.

2.4 Legacy Systems

2.4.1 Problems with Legacy Systems

Despite the fact that the dependency of an entire business software onto the legacy systems make them utmost important and vital; their costly maintenance and enormous source codes are difficult to manage and substitute. Over time, they are losing their compatibility with the current technological mediums and thus getting at the verge of becoming outdated. They owe this loss due to numerous reasons such as inefficiency in drawing and managing the data, unreliability of the fetched data and the challenging acumen needed for their working. As the name implies, legacy systems access and cater data via

the employment of outdated techniques. In order to alleviate the above mentioned drawbacks, these must be substituted with a proficient technical advancement. Apart from these setbacks, legacy systems cannot be completely ousted from the system as they provide the essential support for the information flow [50]. The figure below depicts common issues of legacy systems, which are further covered in detail.

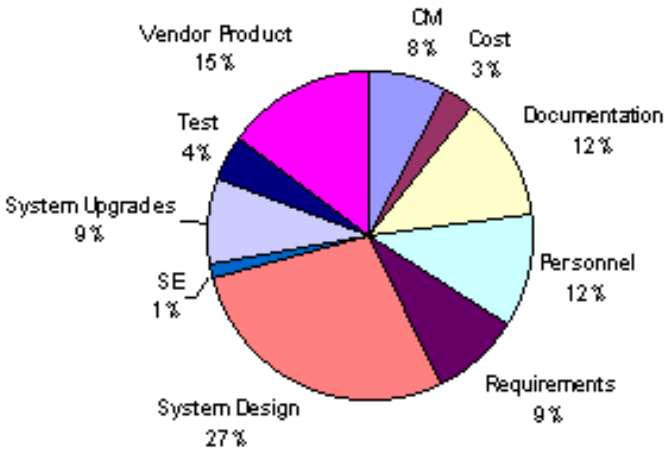


Figure 19: Concerns of Legacy Systems

i) **High Operation Cost:**

Legacy systems call for a high maintenance, operation and supervision budgets. According to a survey, these expenditures can cost up to 90% of the total budget [51], thus, leaving only 10% for other essential operations and hurling the company into a serious monetary concern.

ii) **Maintenance Problem:**

Being an outdated programme, the instruction languages employed in their functioning have also been condemned and are no more part of the present-day curriculum, thus, leaving limited pool of operational specialists. This further adds to the cumbersome idea of modernizing the obsolete software. Moreover, the usage of intricate coding [52] and improper documentation of the legacy systems, only deemed necessary during emergencies [53], make it nearly impossible to preserve and develop them.

iii) **Fewer Experts**

Personnel needed for operating legacy systems are limited, with those available lying in older age brackets and thus ask for higher sums compared to the younger technologists, who are least interested in studying and working in the domain of the old languages' applications, therefore, leaving no option to carry on with the legacy systems. Moreover,

as the field of computer sciences brims up to saturation, students are moving to other disciplines [54].

iv) **Outdated hardware**

Along with outmoded software, there were also little developments on updating the hardware front [55], further adding to the inefficiency of the systems. With more novel technological inventions, the demand for these expensive systems have dropped and therefore, finding a supplier of legacy systems' counterparts is quite a job.

v) **Inadequate Structural Design of Legacy System**

Legacy systems seem to have a chaotic structure and tend to have various gaps in its structural realm. To start with there is no adequate division among user interface and various other prototypes. In addition, they have a stiff and in flexible assembly, thus, offering little to no communion with external hardware and software [56].

vi) **Absence of Proper Documentation**

Yet another shortcoming of the legacy systems is their improper and out of date documentation. Priorly, it could be due to insufficient knowledge and know-how and later because of the retirement of specialists managing these systems. This void caused by lack of paper documentation owing to limited data or structural vagueness have made it more challenging to deal with discrepancies arising in the systems from time to time [55].

2.4.2 Transformation of Legacy Systems

In the recent times, the systems in demand at the enterprises are those which offer high reliability and efficiency along with being handy, inexpensive and flexible. Keeping these factors in view, the sustainability of legacy systems in today's industry seems quite impossible. Legacy systems have long rejected any kind of updates in their software and hardware and as a result are now too obsolete to get in sync with the latest technology and thus, to be employed in organizations.
The above mentioned discrepancies in the legacy systems asks for a major revamp, as the inefficient legacy systems are too slow to respond to the prompt market variations and thus, are a big disappointment on this front. Therefore, to employ them in the industry its essential that they must be upgraded to effectively and efficiently respond to corporate needs [56].

Risks of Legacy Modernization:

Complexity, conformity, changeability and invisibility are respectively identified as the four fundamental measures to building software. Due to the inadequacies mentioned earlier, i.e. firm structural layouts, improper documentation and limited field experts, legacy systems are unable to attain complexity and changeability.

When working on modernizing a system, two fronts need to be catered simultaneously; the technical and the non-technical. The technical part deals with parameters such as usability, software improvement facility and sustenance, safety, information transfer, code preservation and control, approach for migration procedure attainment, etc. The non-technical aspect looks at the humanistic issues such as fear to accept the change and adopt or test new approaches and learn software. Other factors such as costs involved in purchasing new tools, personnel training of novel techniques, etc. may also hamper the updating process [65].

Legacy systems are found all around the globe and are interlinked. Therefore, a change in one system calls for a change in all others, which is a major hamper in their up- gradation and can cause a gap in the working of the organization [66].

The complex structure of legacy systems make it difficult to adapt to the advance technology and practices used in the modern industry. The major issue with the legacy systems is its compatibility with the modern software's, which in turn give rise to the integration of these legacy system in to advance system [66]. Legacy systems made use of traditional languages which are now no more in use and thus, have very limited interpreters which makes it almost impossible to reinstate the language as the respective code understanding is inadequate. Further adding to the list of the hurdles in legacy systems' language conservation is the fact that no proper documentation exists and also there are no records of any up-gradation, if ever done. Therefore, it is near to impossible for enterprises to source out any data from the legacy systems [66].

In a gist, all the above discussed issues make it impossible for the modern day organizations to embrace the idea of legacy system up gradation. Despite of the aforementioned problems professionals seem least interested in replacing or modernizing the legacy systems. The fundamental reasons (as per CIO Insight Magazine survey, 2002) for the faced resistance are as the documentation of the legacy systems, replacement of system and financial resources required in skilled training, difficulty to shift in terms of temporary discontinuing the corporation's tasks and the threat of bearing the failings of replaced or novel system.

The reasons above point to the fact that alongside with the technical aspect, the cultural aspect of the organization has a prominent role to play too. Research studies have verified the stance that cultural environment and perceptive outlook of an organization bids a higher obstacle to modernization as compared to the technical intricacy. Therefore, the human administrative facet is of far more essence compared to the technical facet in making the progress of software effective [67].

Modernization of Legacy System

There are numerous techniques employed for the purpose of achieving legacy modernization and the implementation of these methods may vary from one organization to another as the process is dependent upon a number of parameters. The appropriate approach is needed to consider in the legacy systems' complex design, monetary limitations, prof-

its, syncing of the legacy system with the freshly introduced devices, etc. Therefore, legacy system renovation involves both the technical and the business phases of an enterprise [67]. It spins around the organization's fiscal status and the optimized plan layout which tells what and how modernization has to be done.
The procedure of modernization is said to be comprising of three fundamental factors, namely, market forces, corporate strategies and prudence tactic that plan an overall scheme benefit based on price, profit, risk and flexibility.

2.5 Production Systems

This section discusses and makes a brief analysis of the production system. To understand the production system properly, it is important to understand the meaning of production first. Production can be defined as the process of producing goods or providing services by using combination of capital, material and work. In addition to that, anything from consumer good production, consultancy company, energy production and from music can be considered as production. Creating products and providing services are two important aspects of production. Therefore, it is very important to make a connection between both, because production of goods is useless if it is not combined well with production of services. [57]

Production is defined and discussed above, which can be helpful to understand production system. Production system is defined as the system which converts demand information into products [58]. The system comprised of many resources which are humans, machinery, building and warehouse [58]. Following figure depicts conventional view of the production system.

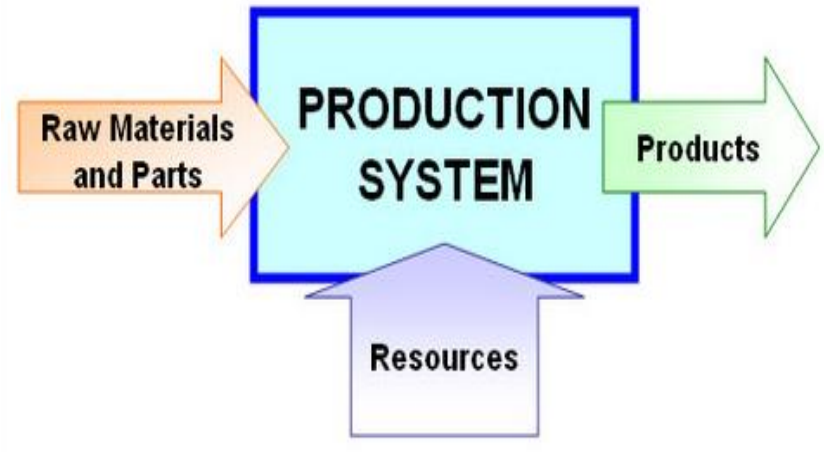


Figure 20: Production System [58]

Further in this section automation and its objectives in production system will be discussed.

2.5.1 Automation in Production Systems

According to [59], Automation is always considered as an effective way to minimize human effort and the production cost in the production system. Other than production processes automation is useful in different processes like transportation, storage and material handling. Moreover, it also provides solutions in highly time critical circumstances where it is difficult for the human operator to respond. According to Satchell (1998) Automation can be defined as the process of replacing human activities with machines activities. [59]

The objectives of the automation in production system are explained below:

i. Maximizing System efficiency:

Automation helps to increase the efficiency of the production system by making the system more flexible. It reduces the human activities to minimal by maximizing the machine activities due to which system can work without any delays which in result improves the overall efficiency of the system. [60]

ii. Improve Quality of product:

One of the advantages of automation is that it improves the quality of products. What automation helps to achieve is that, it minimizes the production errors because it doesn't involve human in any production process and all the work is done by machines. Consequently, it improves the product quality. [60]

iii. Better goods Management:

As automation is considered as revolution in production processes. It also helps to manage the end products quite efficiently. Nowadays, everything is being done online which helps to maintain the record of end products and inventory at the same time. [60]

iv. Information Management:

As the data of product manufacturing is available online so it becomes easier to keep the customer informed about production, dispatching and delivery of product. [60]

v. Improved Safety:

The use of machinery has made production process safer as compared to the production done by human activities. Safety at the production system is improved by utilizing automation system which consists of alarm system and minimizing the human involvement. If any dangerous incident happens then the safety system issues signal by turning on alarms which helps to minimize the damage. [60]

vi. Keep Track of Defective Products:

Manufacturers can track every product by taking advantage of the information systems in the automation system. If any customer returned defective product the system notifies it. This automation system facilitates to ensure quality and adjust the product according to the customer demands. All this is feasible because of the availability of production number and dispatching identification number of the product in the automation system. [60]

vii. Monitoring System:

Automation system also predicts about error or problem in the system in advance so that consequences can be nullified. As automated production system also consists of a monitoring system, which identifies the system failures and sends warning about the predicted defects of the system. With the help of the monitoring system runs without any breakage and delays. [60]

2.6 ISA 95 Standard

2.6.1 ISA 95 Standard

A large missing gap has been observed in the information systems at various data flow stages. Primitively the information systems were developed to be employed in maintaining accounting and stock supervision of the corporations but afterwards their continual advancement allow them to be used in the manufacturing industry and production management. Thus, the systems relied onto the most recent production data. All the various forms of companies' resources ranging from sales to financial and marketing to human were handled by the promising Enterprise Resource Planning (ERP) systems. Despite the fact that these systems were mainly designed to manage and monitor monetary issues, their employment in the production processes consequence in developing an enormous disparity between ERP and automation process control systems, thus, causing an increase in the information management issues relating to assessment of data quality features such as timeliness, accuracy, consistency, etc. [24]

Amid the recent innovations in technology and introduction of digitalized systems, we have found a leeway to extend information to both the process industry and automation softwares, which was formerly not possible, and thereafter, was generating challenges and obstacles in the integration and interfacing of the diverse systems applied in business procedures.

The task to get over these hurdles and incorporate the distinct automation fields was handed over to the ISA SP95 committee with a major aim of developing a novel standard which may permit characterization of various modules and hierarchy levels of information system. In the process, it also allows for a reduction in errors and price along with provision of safety, efficiency and reliability and data integrity maintenance during interface execution. [25]

2.6.2 Hierarchy Levels of ISA 95 Model

The main working standards and operation performed in the manufacturing association often pursue similar standards. The hierarchal design of the organization has been characterized in the standard ISA 95 and is exhibited in Figure 21:

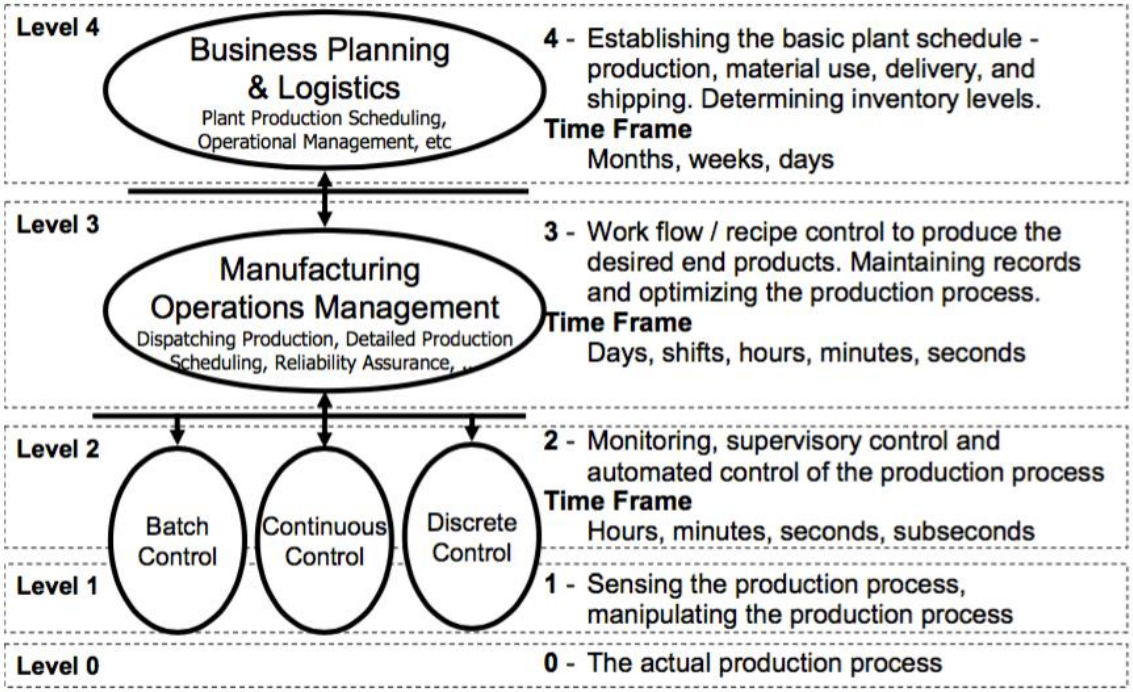


Figure 21: Functional Hierarchy of Automation systems per ISA 95

This hierarchy presents an elementary model for the architecture for the manufacturing systems. The levels characterized in this model plainly characterize and recognize the part and obligations of various units of industry and gives them a reasonable medium and mode for collaboration. It is one of the primary model that established the concept for automation in the manufacturing industry. The automation pyramid demonstrated in Figure 22, describes the flow of information between each level.

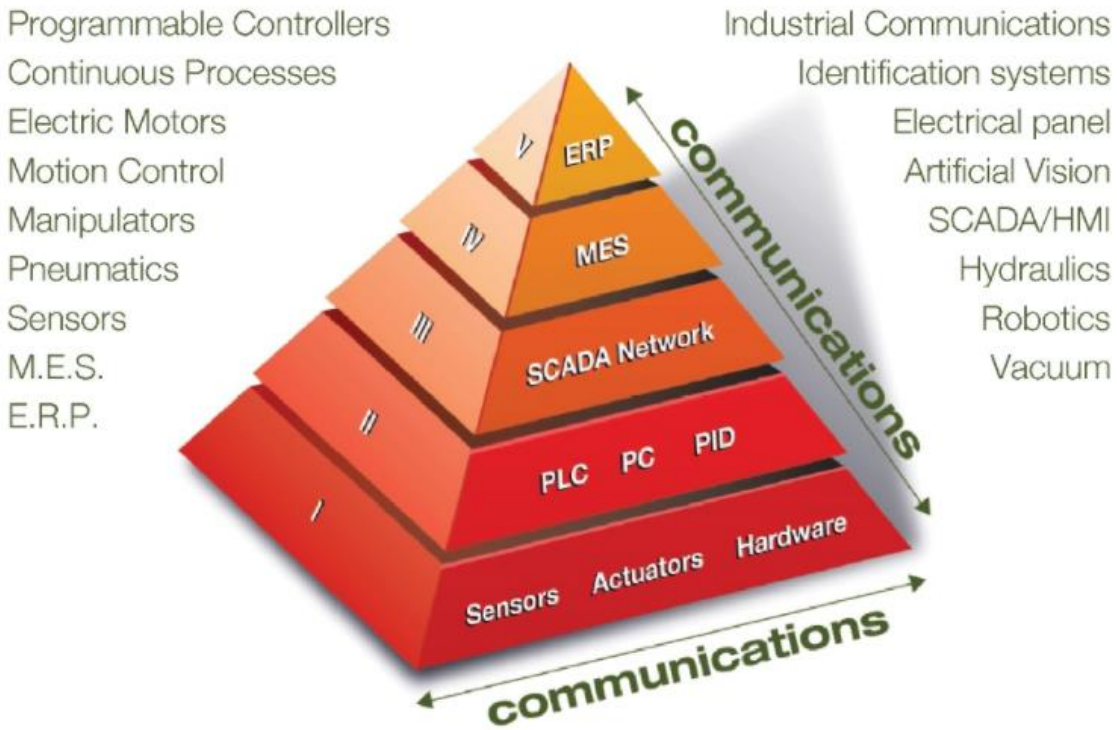


Figure 22: 5-level automation pyramid **[68]**

Level 1, 2 and 3 constitutes of the sensors and process control layer. The SCADA (Supervisory Control and Data Acquisition) and DCS (Distributed Control System) are some of the major sort of process control systems. The bottom three levels consists of the hardware elements, for example, microcontrollers and the electronic circuits. Whereas, the top layer of the pyramid, signifies the business associated activities that are part of the organization. This includes inventory tracking, plant scheduling and logistic services for providing an in time raw material delivery for the production process. The ERP (Enterprise Resource Planning) systems are generally used to automate the processes related to this layer. The MES (Manufacturing Execution System) acts as an integration layer between the lower layers and the ERP layer, by providing the necessary communication and information flow.

The information flow and exchange between the ERP and MES systems is standardized by the ISA-95. The standard consist of five parts.

- Part 1: Models and Terminology. (ANSI/ISA 95.01)

It consists of the mutual terminologies and models for information exchange, used in the manufacturing systems from the top level to the factory floor.

- Part 2: Object Attributes. (ANSI/ISA 95.02)

It defines the attributes of the objects defined in the part 1. It uses the UML (Unified Modelling Language) to elaborate the object model for information exchange.

- Part 3: Models of Manufacturing Operations. (ANSI/ISA 95.03)

It emphasizes the functionalities that are involved in the level 4 of the pyramid (MES layer). The layer is additionally categorized in to the maintenance, quality, inventory and production layers.

- Part 4: Objects and Attributes for Manufacturing Operations Management Integration. (ANSI/ISA 95.04)

This part focuses on the level 3 of the pyramid demonstrated in figure 22, by providing the description of the information flow models.

- Part 5: Business to Manufacturing Transactions. (ANSI/ISA 95.05)

It emphasizes on the transaction activities involving business to manufacturing flow, in the fifth part of the pyramid.

2.7 Function Blocks

2.7.1 IEC 61499 Standard

The novel introduction of this standard is looked at as a significant development in the field of distributed control systems [20]. Likewise, its predecessor, IEC 61131 is employed for the working of the Programmable Logic Controllers (PLCs). IEC 61499 constitutes a type of a Functional Block; a model of assorted, autonomous mechanization units and their mutual interaction. The enhanced feature in the advance version is the user adaptability to attain precise data regarding the various computational divisions. Thus, it aids in rigorous and handy accession of data components [21].

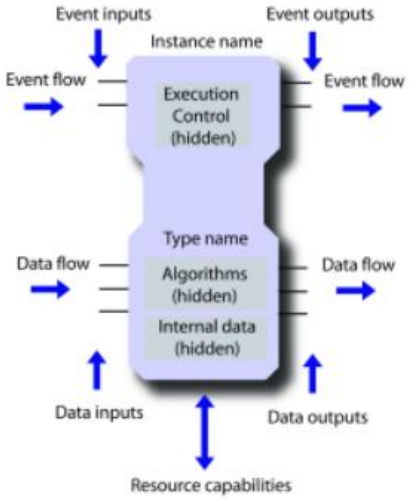


Figure 23: Function Block Model [61]

Each Functional Block consists of two fundamental units. In the figure above, each part has its own specific functions; one handles the controlling mechanism while other deals with the data. The incoming events are fed in to the function block as seen in the block diagram. Next, the controlling unit examines these and in relation to the present state of the function block decides whether to drive them outside or to hold on to them.

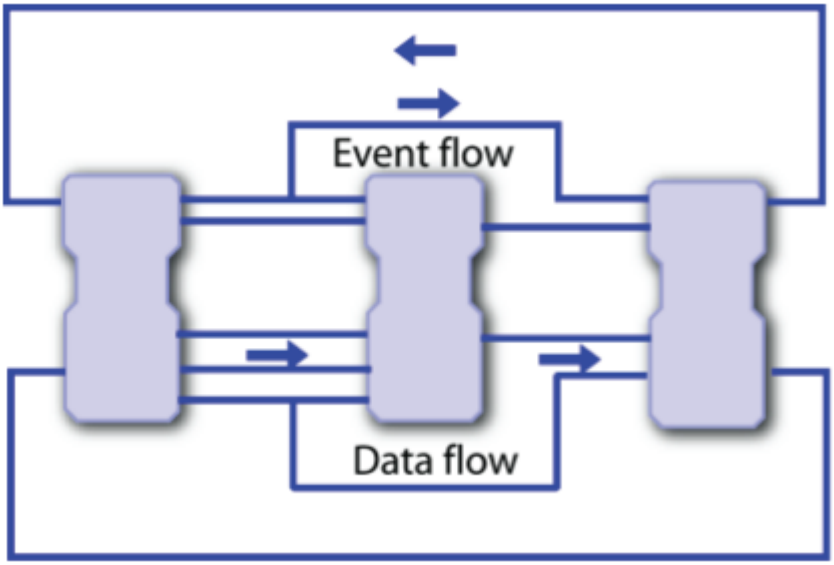


Figure 24: Function Block Interconnection [62]

Finally, the computational outcome permits the Functional Block to take in data from left and to transfer it towards the output units on the right to be employed by successive Function Blocks, which are interconnected as shown in the figure above. [22]

2.7.2 PLANT-COCKPIT

The major aim of the Plant Cockpit project (2013) is to overcome the difficulties faced in incorporating two odd systems by providing an optimized bridge between them [23]. The PCP in accordance with the IEC 61499 standard adopts the function block approach to accomplish the aforementioned purpose.

Previous Implementations:

- *SQL Data Function Block*

The data from relational databases is carried by the SQL Adapter, with the aid of Structured Query Language (SQL), which accepts the user given input configurations in the JSON format. These configurations usually include three key components: the headers, the sources and the output schema. The main function of a header is to recognize the created request of adapter via various ids. Whereas, the relational database's data related sort, site and verification is saved in the source. Lastly, the task of providing the precise information to the operator to restore the specific data is done in the output schema, which employs the JSON configuration to serve the purpose. Various databases can be put up and catered using such an adapter, e.g. the existing edition can successfully be employed along MariaDB, PostgreSQL and MySQL.

- *Excel Sheets Data Function Block*

Keeping in view the need of legacy systems, which have most of their data in form of excel sheets, an Excel adapter is designed to attain the required data from various excel files. These excel files can be accessed remotely on the internet with appropriate validation (if they are being hosted on an FTP server). The Excel adapter has the same arrangements except for the sources, which now contains the file name, FTP server and certification specifications while the output schema holds the cell references to the excel file. The data in the output schema of this adapter also proceeds in the user desired JSON format.

3 METHODOLOGY

This chapter will illustrates the phases this thesis has gone through and the research methodology adopted to solve the identified problem in the light of the background knowledge presented in the previous chapter. This chapter will also present the overall deployment architecture as well the flow diagram.

3.1 Approach

The initial part before commencing the implementation was to create a general diagram of the overall system, in order to grasp the overall concept of the framework. As can be seen in figure 24, a general diagram was generated to conceptualize the general idea of the project.

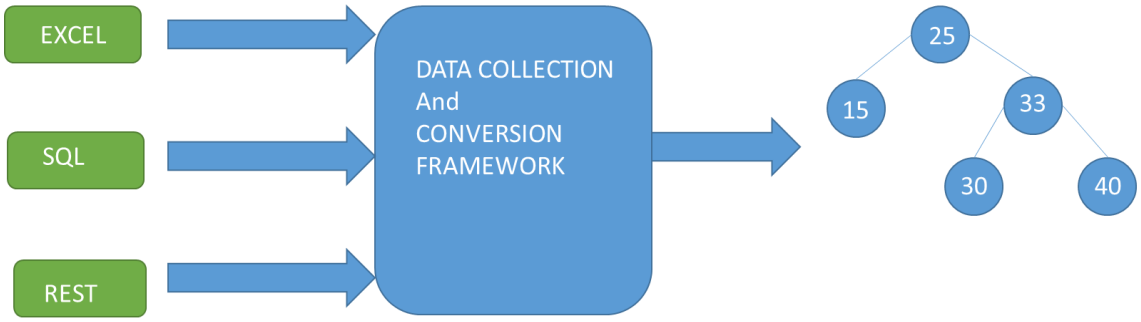


Figure 25: General System Overview

The thought behind the formation of this diagram was to consider the system like a black box, with specific input sources, which yielded a specified output. Once it was clear what the inputs and outputs of the system must be, a more detailed diagram was made, reflecting the flow of information between each source. The primary objective of this diagram was to further break down the generalized system in to more specialized block and thereafter, assigning specific task to each block. Thereafter, a general interaction strategy was devised as how these blocks must communicate, in order to achieve the desired objective. Figure 26 demonstrated a fragmented view of the generalized framework presented in Figure 25. Figure 26 demonstrates the major processes involved in the framework, which are as follows:

i) **Fetch Data:** The first major step is concerned with the gathering of data from the specified data source. The data can be fetched from multiple data sources, and will be displayed to the user in the HTML format.

ii) **Map Data:** Once, the user understands the data; they can specify the fields to be selected from the source. Additionally, the user should specify the desired data source to map the acquired data.

iii) **Analyze Data:** Once, the above two processes are completed, the user can benefit from the frameworks ability to query and analyze the data source. For example, if the user maps the data to Relational Data Base, they have the options to perform SQL queries on the mapped source.

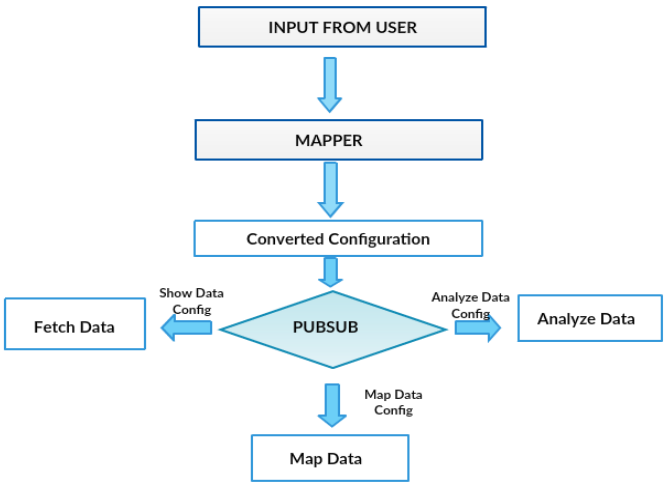


Figure 26: Workflow for the System

The diagram shows the major responsibilities of the PUBSUB block, specifying how it manages the above-mentioned processes. Once, the major components and their respective interaction was apparent, the next part was to design the detailed architectural overview, highlighting the specific blocks and technologies used. The following section will present the architectural overview of the system.

3.2.1 Architectural Overview

The architecture diagram is a basic framework that is often used in the phase of system planning. It helps in explaining the architecture, communicating the ideas clearly and discussing any required changes. Thus laying the ground for more operational and elaborated functional architecture of the solution. The architectural diagram for the system can be viewed as follow:

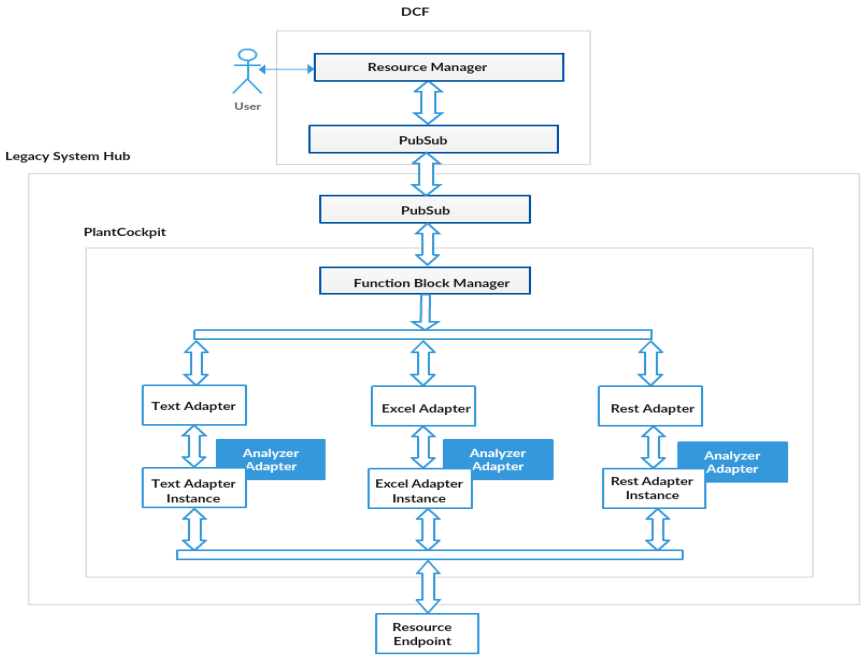


Figure 27: System Architecture Diagram

The diagram demonstrates the components interaction and their respective responsibilities in achieving the expected result. It visualizes the creation of Adapter Instances used for fetching the data from the desired sources and thereafter shows how the created instances communicate with the Analyzer Adapters to insert and query the data.

The process in divided in to four major components involving:

i) Data End Points
ii) Legacy System Hub (LSH)
   I. PubSub
   II. Mapper
   III. PlantCockpit
iii) Analyzer Adapters (AA)
iv) Data Collection Framework (DCF)
   a. Resource Manager(RM)
   b. PubSub

Data End Points comprises of three sources, from which the user can fetch the data to be converted, i.e.: i) EXCEL files ii) TEXT files and iii) HTTP endpoints. Legacy System Hub consists of the *PubSub module*, *Mapper* and *PlantCockpit*. The *PubSub* module is used for communication between the *LSH* and *DCF*, whereas *Mapper* is used to convert the high level input from the user to the desired JSON configuration. *DCF* consists of *Resource Manager* and *PubSub* module. *RM* is responsible to take users input configuration and send them to the *LSH* using the *PubSub* module. Once the *Mapper* parses the input, it is sent to the *Function Block Manger* (*FBM*) for creation of the new adapter

instance. After the adapter's creation, it further creates the respective *Analyzer Adapter* as per the user's requirements. The created instance continuously checks the data source for modifications, and updates itself as soon as the respective source is updated. *Analyzer Adapter* is responsible for creating the requested table/schema or data structure and inserting the cleaned data in to the respective storage. It is also responsible to provide respective processing or analyzes on the inserted data.

The following block diagram explains the creation of adapter instances and their respective interaction with the analyzer adapter.

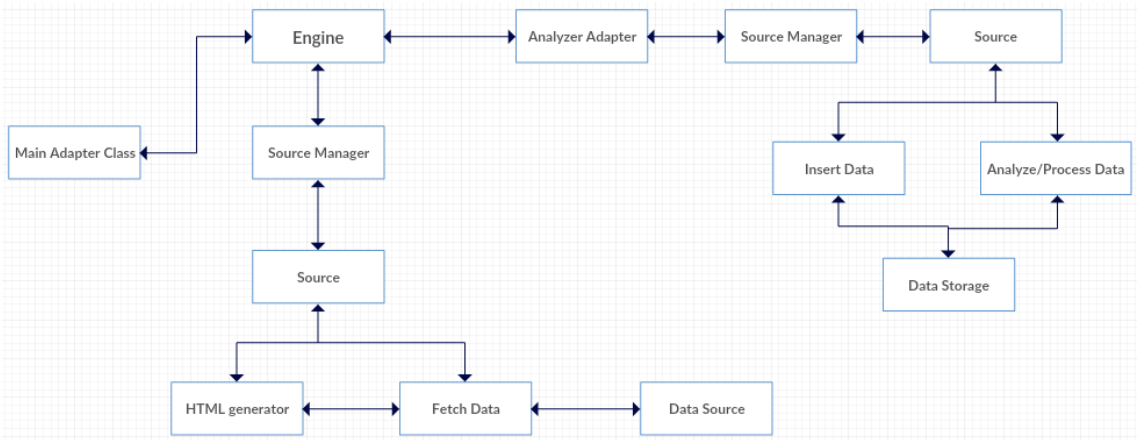


Figure 28: Illustration of Adapter Instance & Analyzer Adapter

After the *FBM* creates the new instance, the configuration for the data source is sent to the Main Class of the Adapter. The *Engine* is thereafter responsible for managing the respectable responsibilities of the Instance. Firstly, the *Engine* creates a *Source* object (based on the configuration), and thereafter uses *Source Manager* to store the relevant source in a hash map against their source ids. The *Source* is responsible for fetching the data from the *Source* and to generate the HTML. Moreover, *Engine* keeps a check on the data source and fetches the updated data as soon as it is available.

In addition to all this, the *Engine* send the configuration to the respective *Analyzer Adapter*, which further contacts its *Source Manager* to create the specified type of source. The *Analyzer Adapter* is thereafter in charge of inserting, updating and analyzing/processing the required data format.

3.2 Tools and Techniques

ANTLR

As the name suggests, ANTLR (ANother Tool for Language Recognition) is yet another parser tool for processing, reading and translating the structured text. The parser works by taking the string input and thereafter transforming it in a more structured format, such as an AST (Abstract Syntax Tree). In order to obtain this organized structure, ANTLR

first needs a grammar in order to generate a lexer and a parser. Thereafter, the generated parser is used to build and then parse the AST.

Apache Service Mix

It is an integration container unifying numerous functionalities provided by Apache Camel, ActiveMQ, Karaf and CXF. This powerful platform thus provides an ESB (Enterprise Service Bus) established on the very essential principles of SOA (Service Oriented Architecture). The platform solves some of the major issues faced by the IT industries, i.e: data exchange between diverge systems, routing of messages between different systems and integration functionalities. Moreover, the Service Mix ESB adheres to the standard laid down by the Java Business Integration (JBI). This thesis is built upon Apache Service Mix, hosting the *PlantCokcpit* project to deploy the respective Adapter bundles. Service Mix manages the lifecycle of these bundles, so they could be used for hot deployment. In addition to this, it further checks and verifies the dependencies required for each bundle to work correctly.

PlantCockpit

PlantCockpit provides an application integration platform, capable of bridging the gap of communication and interaction of the systems. This platform has been used to create the following adapter, i.e. Excel, REST & Text adapters, Linked List Analyzer & SQL Analyzer adapters. The platform has been built on the Function Block specifications as per the IEC 61499 standards, thus providing a loosely coupled system, where components are capable of communicating by leveraging the facilities provided by the ESB.

Software used for development

1) *Eclipse Neon* for development of Legacy System Hub
2) *Atom* for managing the *PubSub* Mockup server
3) *Postman* (Subclient) for checking the endpoints of REST Adapter
4) *Microsoft Excel* for managing the excel file for the Excel Adapter
5) *OpenSSL* for creation of the *SSL* Certificates.
6) *NodeJs* for hosting the *REST* mockup and *PubSub* mockup servers
7) *MYSQL Server* for hosting the database *for SQL Analyzer Adapter*
8) *MYSQL Workbench* for managing the SQL database

4 IMPLMENTATION

This section will explain the solution to the research problem discussed earlier. It provides user the option to conveniently convert data from one format to the other, and thereafter allowing the user to process and analyze data. The interaction between components of the system is described in details in the upcoming section. Moreover, this section also provide some of the use cases for the implemented solution.

4.1 Interaction of Systems

This section will extend the architecture diagram from the previous section to further elaborate the communication between each of them. The first part of this section explains how the mapper is used to validate and then generate the required configuration. Thereafter, the interaction between the *PubSub* and the *FBM* is described and their interaction with the Adapter Instances. Then the interaction between the created Adapter Instances and Analyzer adapters is explained. The last part describes the communication with the Analyzer Adapter for analysis purposes.

4.1.1 Interaction for the Mapper

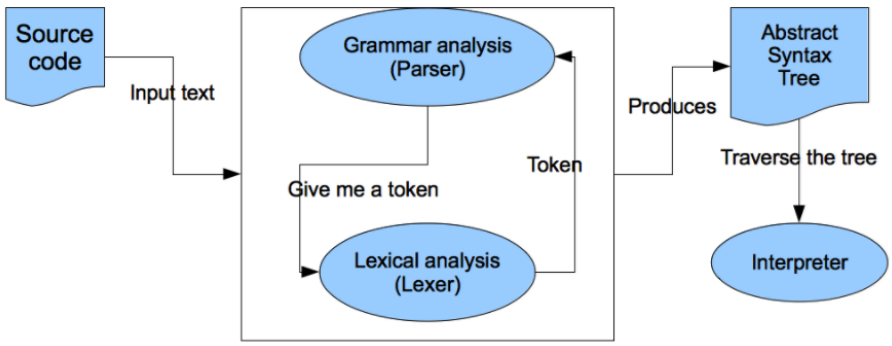


Figure 29: Functioning of Parser to convert High Level Language

Parser takes some input text as input and thereafter builds a data structure in form of an abstract syntax tree. Parsing of the input involves two major steps:

1. Lexical Analysis
2. Syntactic Analysis

Lexical Analysis takes string as a input and produces a stream of tokens, which is thereafter, fed as input to the Syntactic Analyzer, which checks whether the generated tokens either form a correct grammatical meaning or not. This correct grammatical formation of the text leads to the creation of an abstract syntax tree. The generated tree is thereafter

parsed to generate the desired JSON schema. Part of the lexer and parser grammars have been shown in the appendix. It can be seen how the lexer part defines the grammar to generate the respective tokens. Moreover, the parser part defines how the respective tokens should be ordered in order to generate the abstract syntax tree.

An example of the conversion of high level input to JSON can be seen in the following code. It can be observed what were the respective tokens and what should there perceived order be, in order to generate the correct configuration.

Mapper Input:

```
Download[File]moldes cavidades iguales.xls[/File]from[ADDRESS username="user_ad-
min" password="pass_admin"]localhost[/ADDRESS]
```

Mapper Ouput:

```
halfConfig: {
            address: {
                        url:"localhost",
                        username: "user_admin",
                        password: "pass_admin“
                    }
            file: "moldes cavidades iguales.xls"
 }
```

Code 1: Example of Mapping Higher Level Language to JSON

As mentioned in the methodology section, ANTLR has been used to manage the lexer and parser grammars. The first step while using the ANTLR software is the creation of .g4 extension files, which defines the rules of the language that is under analysis. Once, these files are finalized, the antlr4 program will be used to create the files, which will be used by our software (e.g. the lexer and parser). Firstly, a target language is specified in order to generate a parser (JAVA language was chosen for generating the parser). Thereafter, the visitors and listeners are generated by the using the antlr4 program.

The following section will explain how the respective grammar for the framework has been created and how to incorporate new grammar in to the system. Additionally, it explains the process of creating the JSON configuration by parsing the abstract tree generated by the antlr4 program, based on the defined grammar.

a) Creating Grammar:

- **Lexer Grammar:**

First, we start of by using the fragment rules. They are reusable building blocks of the lexer rules for making the grammar more readable and easier to maintain. Fragments are only supposed to make the lexer grammar easier to use, and therefore, do not qualify as valid tokens. In the implemented lexer grammar, the fragments have been defined for the letters that were frequently used in the keywords. Moreover, for this implementation, the

grammar has been kept case insensitive and therefore the fragments for the alphabets are defined as follows:

```
fragment A          : ('A'|'a') ;
fragment B          : ('B' | 'b') ;
fragment C          : ('C'|'c') ;
```

Code 2: Defining Fragments

The above grammar was replicated for all the alphabets from A to Z, thus making the grammar easier to use. The other major fragment is the "LETTER", which indicates a single case insensitive alphabet. The token "ID" is derived from the "LETTER" fragment and could be a combination of multiple "LETTER", as shown below.

```
fragment LETTERS    : [a-zA-Z] ;
ID                  : LETTERS+ ;
```

Code 3: Usage of Fragments for facilitating lexer tokens

This "ID" token, will be used ahead in the parser grammar, in order to define the desired grammatical formation of the tokens.

Additionally the lexer grammar has been set to the "BBCODE" mode, in order to make the language more formal, by taking advantage of the already defined rules, thus helping to prevent extra effort for the grammar. Thereafter, the tokens, which are to be used in the parser grammar, are all defined in the lexer grammar.

Note: Besides these above mentioned example, more lexer grammar rules are defined in the APPENDIX A – LEXER AND PARSER GRAMMAR.

- **Parser Gramar:**

The next step after the creation of the lexer grammar is to define the parser rules for the grammar. The first and foremost part before writing the rules for the parser is to tell the parser which lexer rule should it be using. This is accomplished by using the options tag in the parser grammar, as shown below:

```
options { tokenVocab=MarkupLexer; }
```

Code 4: Options Tag

The next step after defining the desired lexer file, was to define the attributes which will be used for different data formats (e.g. for text adapter, REST adapter etc.).

```
driverattribute         : DRIVER '=' STRING ;
addressattribute        : ADDRESS '=' STRING ;
schemaNameattribute     : SCHEMANAME '=' STRING ;
usernameattribute       : USERNAME '=' STRING ;
passwordattribute       : PASSWORD '=' STRING ;
tablenameattribute      : TABLENAME '=' STRING ;
filenameattribute       : FILENAME '=' STRING ;
restnameattribute       : RESTNAME '=' STRING ;
pathnameattribute       : PATHNAME '=' STRING ;
idattribute             : IDNAME '=' STRING ;
nameattribute           : NAME '=' STRING ;
separatorattribute      : SEPARATOR '=' STRING ;
```

Code 5: Defining Parser rules

The left part of the code defines the parser rules, whereas the right part defines, how these rules must be used, and therefore provide the desired sequence of tokens for each one of them. For example, the "driverattribute" rule will expect the data to be arranged in the following format:

```
driver=”com.mysql.jdbc.Driver”
```

Code 6: Example of Usage of Lexer Rule

After defining the attributes needed for various adapters, the next step is to define which attributes are compulsory against each adapter.

```
newsqlquery  : '[' SQLANALYZER idattribute driverattribute addressattribute schema-
Nameattribute usernameattribute passwordattribute ']' content '[' '/' SQLANALYZER
']';
```

Code 7: Defining Attributes for an Adapter

As can be seen in the above code, that the required attributes for the SQL Analyzer Adapter are the id, driver, address, schema, username and password attributes for the respective database. Until and unless these attributes are not available, the parser grammar will report an error to the user to define the compulsory parameters for the respective adapter. Similarly, the grammar for each of the Adapter must be defined in the same manner in the parser rules.

Since, the main objective is to convert data from one format to another; the first part was to define the grammar for the Analyzer Adapters, as shown above. The next part is to define an object, which will hold all the formats the user wants to convert the data. This can be visualized as follows:

```
specsconvert   : '[' CONVERT ']' (newlistquery* newsqlquery*) '[' '/' CONVERT ']';
```

Code 8: Holder for multiple Adapters

This part of the parser grammar signifies a list of Analyzer Adapters that the user wants the data to be converted in to. Notice that all of these Analyzer Adapters have been made optional by making use of the star (*) symbol.

Thereafter, in the next part, the above-mentioned "specsconvert" rule is included inside the "specsforfile" and "specsforrest" rules (the user can define new data type, e.g: excel and can include the "specsconvert" inside it). It can be noticed below, that the "specsforfile" rule, takes the name of the file, username, password and path on the FTP server as its compulsory attributes. Additionally it take the "specsconvert" (defined earlier) and the fields to convert (mentioned by "singfield" rule) as its content.

```
specsforfile  : '[' FORFILE nameattribute usernameattribute passwordattribute  path-
nameattribute separatorattribute ']' singlefield* specsconvert '[' '/' FORFILE ']';
```

Code 9: Parser Rule for converting text data

The "singlefield" rule, defines the properties of the respective fields to be added from the data source. This normally includes the row and column offset in case of the text and excel files, but only include the field name and type in case of the REST data source.

b) Defining Grammar for New Data Source:

In order to cater the new data formats, the user must specify the lexical and parser grammars accordingly. These are the three steps which user must take to cater new data formats, as shown below in the diagram.

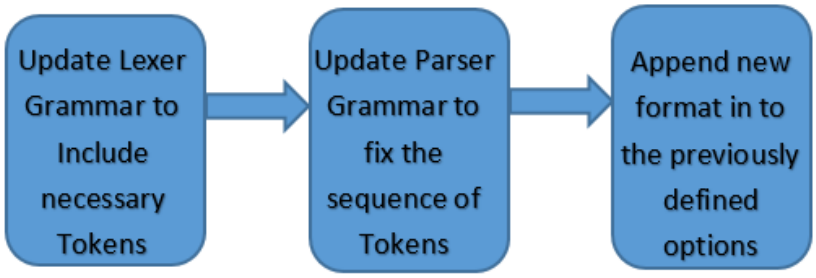


Figure 30: Graphical Representation for Modifying Grammar for new source

The first step to add new data format is to update lexer grammar. For instance if the user wants to introduce Excel convertor in to the system, then the user must specify a new lexer grammar by the name "ExcelAnalyzer" (note: tag name should correspond to the name of the Analyzer Adapter that has been created in the Legacy System Hub, i.e. pid_eu.plant.ExcelAnalyzerAdapter in this case). Thereafter, the user must specify in the parser grammar (as shown below), which tokens will be the part of this new tag and in which order they should be arranged.

```
newexcelquery  : '[' EXCELANALYZER idattribute sheetnameattribute addressattribute
usernameattribute  passwordattribute  pathnameattribute  ']'  content  '['  '/'
EXCELANALYZER ']';
```

Code 10: Tokens sequence for Excel Analyzer

Whereas, token in the current context is defined as an instance of a sequence of characters in specific record that are assembled together to form a meaningful semantic unit for processing. The necessary tokens are the attributes or keywords that must be present in the configuration of the data source (as listed in APPENDIX A – LEXER AND PARSER GRAMMAR). In this case, the *idattribute*, *sheetnameattribute*, *addressattribute*, *usernameattribute*, *passwordattribute and pathnameattribute* are the necessary tokens for the successful parsing of the input. Thereafter, the user must append the new data format, shown in the following code, in the already available options of the system.

```
specsconvert  : '[' CONVERT ']' (newlistquery* newsqlquery* newexcelquery*) '[' '/'
CONVERT ']';
```

Code 11: Appending new Format in to the System

Once, it has been done, the conversion process is taken care of by the framework. The approach checks for the necessary attributes required for the specific Adapters.

In case the users do not want to always specify the grammar for the new data format, they could use a wild entry through the help of the following grammar; however, it does not allow the users to catch errors, since they have not specifically mentioned the grammar for the new source.

```
wildentry     : '[' GENERICANALYZER typeattribute attribute* ']' content '[' '/'
GENERICANALYZER ']';

specsconvert  : '[' CONVERT ']' (newlistquery* newsqlquery* wildentry*) '[' '/'
CONVERT ']';
```

Code 12: Generic Adapter Rules

In the case of wild entry (generic adapter grammar), it can contain multiple attributes, but all of them are optional, and the user can easily omit some of the necessary attributes for the specific adapter type, thus leading to a higher chance of error. Moreover, the generic adapter does not allow more complex and adapter specific logic, and therefore, must be explicitly defined in the grammar tag of the specific source. In case of the generic adapter grammar, the "***typeattribute***" is an essential field. It should be set to the name of the created Adapter in the system (e.g: ***pid_eu.plant.LinkedAnalyzerAdapter***). This is the only way to ensure which adapter the generic grammar is specified for.

c) Creation of Abstract Tree:

Once antlr4 program generates the visitors and listeners from the defined grammar rules, the visitor can be used to parse the abstract tree generated by the ANTLR. A class has been created to extend the "MarkupParserBaseVisitor" class, generated by the ANTLR, in order to traverse the abstract tree generated by it. The abstract tree generated for the following configuration (Code 13), can be seen in Figure 31. The ***part a*** shows the tree for the field tag, whereas the ***part b*** shows the tree for the converted data sources.

```
[forfile name="cambio moldes.txt" username="admin" password="admin" pathname="/RawData" ]
       [field select="MAQ"  rowOffset="0" columnOffset="0" mapTo="MAQ" typ="String"][/field]
       [field select="PIEZA" rowOffset="0" columnOffset="1" mapTo="PIEZA" typ="String"][/field]
       [convert]
                  [LINKEDANALYZER id="20"] order [/LINKEDANALYZER]
                  [SQLANALYZER id="22" driver="com.mysql.jdbc.Driver" address="mysql://localhost/"
                         schemaName="TestDB" username="root" password="root"]order[/SQLANALYZER]
       [/convert]
[/forfile]
```

Code 13: Sample Configuration for Converting Text data

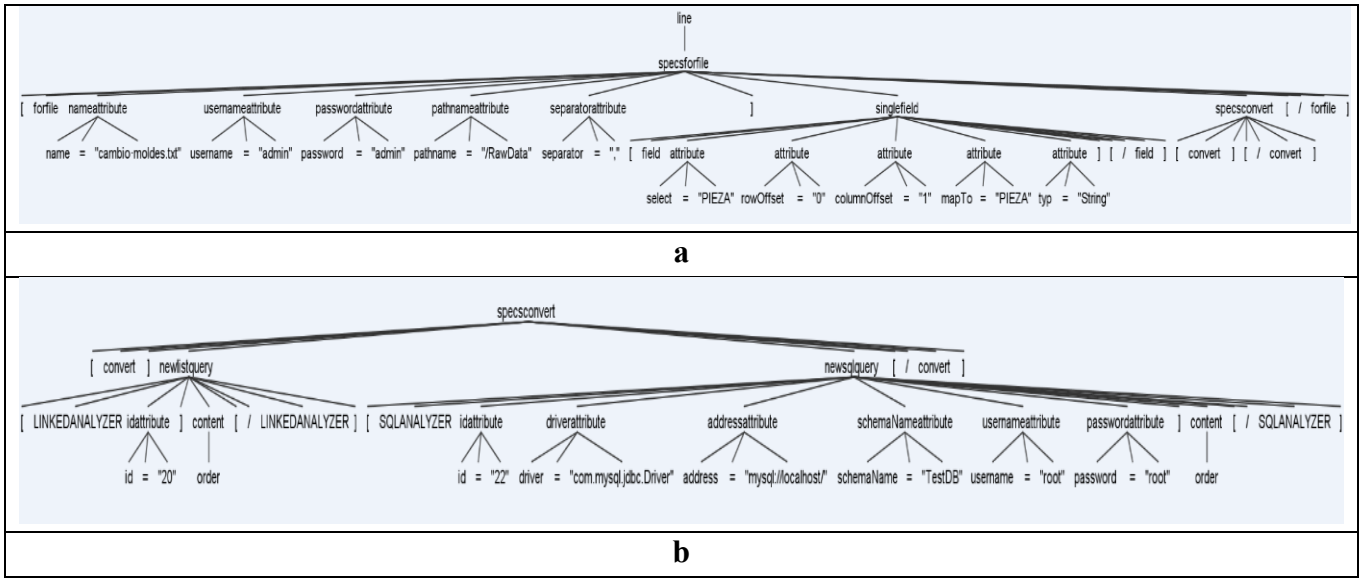


Figure 31: Abstract Tree for Configuration

Now after the creation of abstract tree, the *interpreter* comes in to action (as shown in Figure 29). While traversing the abstract tree, the *interpreter* analyzes all the field tags that are present in the input and their respective attributes are stored to form the fields Array, against all their specified attributes (Code 14). Whereas, the interpreter is programmed in a way that the configuration for the SQLANALYZER or LINKEDANALYZER (as highlighted in Code 11) are also stored, along with their necessary attributes. Following is an example, which demonstrates, how the fields against a text file, are stored along with their attributes in to an Array. The process for storing the configuration of the ANALYZER ADAPTERs is almost the same as those of storing the fields attributes.

```
1. @Override
2.     public void visitSinglefield(MarkupParser.SinglefieldContext
   context){
3.         String text = "";
4.         List<TerminalNode> ids = context.FIELD();
5.         if(ids.size() != 2)
6.             throw newError("ENDING TAG NOT DEFINED.....");
7.
8.    List<MarkupParser.AttributeContext> allAttribute =context.attr
   ibute();
9.    JSONObject jo = newJSONObject();
10.   for(MarkupParser.AttributeContexttemp : allAttribute) {
11.      String val =temp.STRING().toString().replace("\"", "");
12.      jo.put(temp.ID().getText(), val);
13.   }
14.
15.   for (MarkupParser.ElementContext node: context.element()){
16.       if(node.content() != null){
17.          String val = visitContent(node.content());
18.          jo.put("name", val);
19.       }
20.   }
21.
22.   fields.put(jo);
23. }
```

Code 14: Managing Fields via Interpreter

Once, the antlr4 program finishes the processing of the abstract tree, the output configuration from the antlr4 program is fed in to the PubSub module, which then uses it to generate the Adapter Instances and Analyzer Adapters as well. The process for creating the Adapter Instances and Analyzer Adapters is discussed in the upcoming sections.

4.1.2 Interaction for Adapter Instance

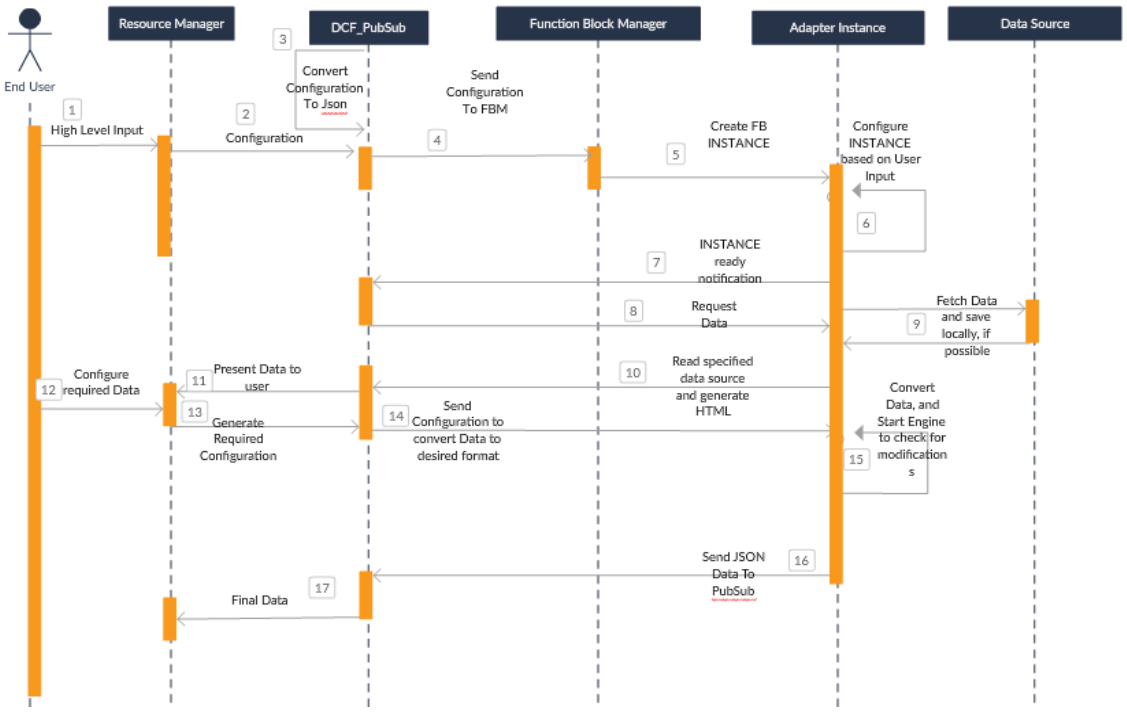


Figure 32: Adapter Creation & Fetching of Data

The user passes the input as text format from the user interface *RM*, which is then sent to the *PubSub* module. Upon receiving the text, *PubSub* module takes advantage of the *MAPPER* and converts the high level input to the respective JSON configuration. The JSON is there after sent to the Function Block Manager (FBM), which creates the new adapter instance based on the settings provided by the user. The newly created instance is configured based on the JSON fields, and a notification is sent to the *PubSub* module notifying the correctly configured instance. Upon receiving the notification, the *PubSub* module enquires the *Adapter Instance* for the html format of the data. As soon as the HTML is received from the *Instance*, it is sent back to the *RM*. Thereafter, the end user monitors the data from the corresponding data source and determines which fields he wants to select. In addition to the fields, the user also selects the output data format and the fields to map the data. Upon receiving the text input, the *PubSub* again uses the *MAPPER* to check and convert the text in to the required configuration. The configuration is sent to the *Adapter Instance*, which fetches the specified data from the source and thereafter convert it to a JSON Array. The converted JSON is sent back to the *RM* and to the respective *Analyzer Adapter* as well. (Sequence diagram explaining the functionality of *Analyzer Adapters* is explained in ***Figure 33***.). After being fully configured, the Adapter Instance, queries the data source (every 10 seconds) to check for any modifications. In case, the data is updated it is resent to the *RM* and the *Analyzer Adapter*.

4.1.3 Interaction for Analyzer Adapter

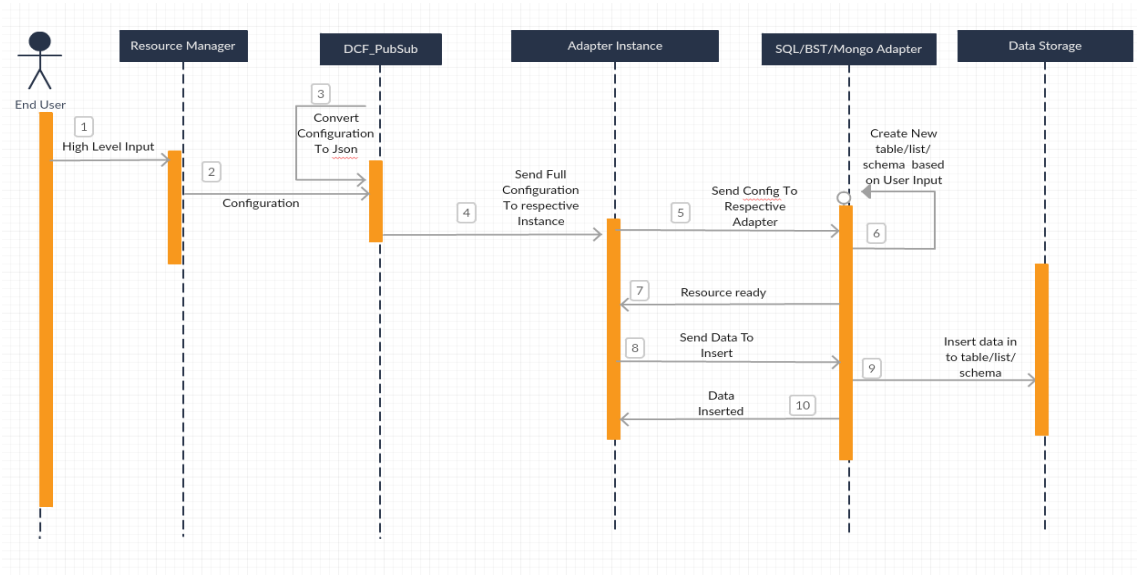


Figure 33 Creation of Data Storage in the Analyzer Adapter

The above diagram further explains the communication between the *Adapter Instance* and the *Analyzer Adapters*. As soon as the *Adapter Instance* fetches the data and convert it to the JSON object, it is also sent to the corresponding *Analyzer Adapter* as well. Upon receiving the input, the *Analyzer* creates the specified data structure/table or schema, and notifies to the *Adapter Instance*. Thereafter, *Adapter Instance* asks the *Analyzer* to insert the data in to the relevant data storage. Since, data is stored, in the desired format, the user can later, query the data storage to get the processed output. The following sequence diagram (***Figure 34***) demonstrates the flow between components, in order to analyze/process the data.

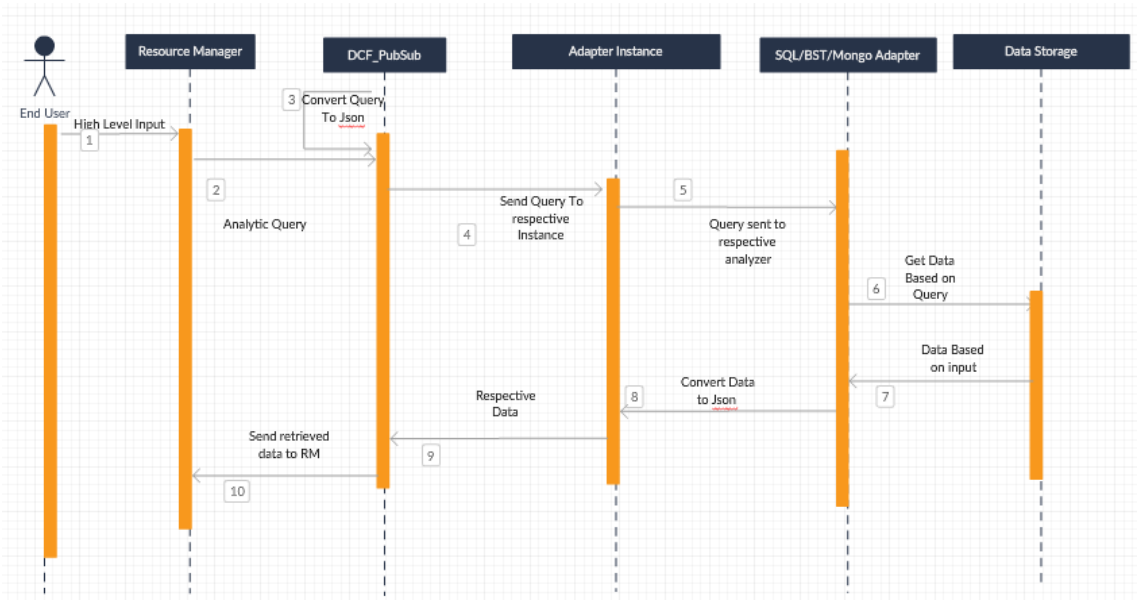


Figure 34 Querying of Data Storage in the Analyzer Adapter

The user passes the input as text format from the user interface *RM*, which is then sent to the *PubSub* module. Upon receiving the text, *PubSub* module takes advantage of the *MAPPER* and converts the high level input to the respective JSON configuration. The JSON is there after sent to the *Adapter Instance*, which upon receiving the query passes it on to the *Analyzer Adapter*. The respective query is processed and the output is converted in to a JSON format, which is sent back to the *RM*.

4.2 Use Cases

The use cases of the created *Adapters* and *Analyzers* is presented in this section. It discusses an in detail implementation of each of the implemented *Adapters* and the *Analyzers*.

4.2.1 Text Adapter

Text adapter is used to acquire data from the text files. The first step in the implementation of the Text Adapter is the availability of the FTP server to fetch the files from. In order to retrieve the respective file, the corresponding credentials, path of the file in the server and name of the file must be mentioned in the configuration. Sample configuration for the LSH is shown below:

Mapper Input:

```
Download [File]moldes cavidades iguales.xls[/File] from [ADDRESS username="user_admin"
password="pass_admin"]localhost[/ADDRESS]
```

Example of Input Configuration to LSH

Upon receiving the Input configuration, the *FBM* creates and configures the *Adapter Instance*. The *Instance* then uses its *Engine* to fetch the data and display the respective HTML to the user. The following image, displays the HTML data, from the data source. Thereafter, the user analyzes the data and selects the desired columns and other relevant features.

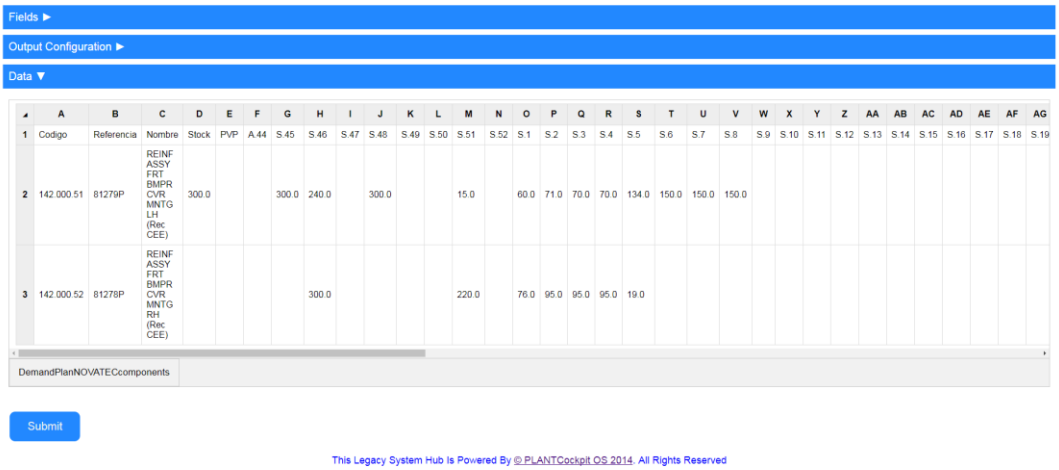


Figure 35 Resource Manager displaying fetched data as HTML

The full configuration received from the *RM* is then forwarded to the Adapter Instance. An example of the full configuration can be seen as following:

Mapper Input:

```
[forfile name="cambio moldes.txt" username="admin" password="admin"
pathname="/RawData" separator="," ]
        [field columnOffset="0" mapTo="MAQUINA" typ="String"][/field]
        [field  columnOffset="1" mapTo="PIEZA" typ="String"][/field]
        [convert]
                [SQLANALYZER id="22" driver="com.mysql.jdbc.Driver"
                address="mysql://localhost/" schemaName="TestDB"
                username="root" password="root"] order [/SQLANALYZER]
        [/convert]
[/forfile]
```

Figure 36: Example of Full Configuration to LSH

While implementing the text adapter, it is assumed that each data row in the file is separated by new line character. Additionally, the user must pass the separator string as a mandatory input, since it is responsible for separating each row in to multiple columns based on the separator value.

As soon as the full configuration is received, the *Mapper* parses the input configuration and checks for all the necessary fields. It is converted to the desired JSON as per needs of the Text Adapter. The data is fetched from the specified text file and is converted to the JSON Array based on the *field* parameters defined by the user. The converted JSON is then forwarded to the corresponding Analyzer Adapter, together with the desired schema. Following interface of the SQL Analyzer Adapter can be seen in the following diagram.

```
package eu.C2NET.ManageSQL.Interface;

import org.json.JSONArray;
import org.json.JSONObject;

import eu.C2NET.ManageSQL.Manager.MainAdapter;

public interface SqlService {

	public void addSource(JSONObject sourceSchema,  JSONArray
dataToInsert);
	public JSONArray querySource(String sourceID, String inputQuery);

}
```

Figure 37: SQL Analyzer Adapter Interface

The "*addSource*" function is responsible for creating the table schema based on the configuration provided by the user. It also receives the data to be inserted in to the table, as one of its input. It can be further seen that the "*querySource*" demands the query to be executed against the table. This function is responsible for returning the response of the query back to the user.

Once, the data is fetched, cleaned, converted and saved in to the data base, the user can pass on the queries to analyze the respective data. The user can pass the configuration for analysis to the system as depicted in Figure 34.

The analyzing configuration received from the *RM* is forwarded to the Adapter Instance. An example of the analyzer configuration can be seen as following:

Mapper Input:

```
ANALYSE
[RESOURCE id="11" analyzerid="22"]
SELECT * from testdb.order where testdb.order.MAQUINA = "2/400"
[/RESOURCE]
```

Figure 38: Example of Analyzer Configuration to LSH

As soon as the configuration is received, it is converted to the desired JSON. The adapter instance, checks for the available analyzer adapters, and checks for the requested adapter. If the requested adapter is available, then the respective input query is forwarded to the SQL Analyzer Adapter. Once, the Analyzer Adapter receives the query, it processes the query and thereafter, produces the dynamic JSON array based on the results of the provided input. This JSON array is then sent back to the *RM*.

4.2.2 REST Adapter

REST adapter is used to acquire data from various REST endpoints. The first step in the implementation of the REST Adapter is the availability of the HTTP server to fetch the data from. In order to retrieve the respective JSON Array, the corresponding credentials and name of the resource must be mentioned in the configuration. Sample configuration for the LSH is shown below:

Mapper Input:

```
Download [REST]ORDER[/REST] from [ADDRESS method=”GET”]
http://localhost:3001/merphelper/api /Order/all  [/ADDRESS]
```

Figure 39: Example of Input Configuration to LSH

Upon receiving the Input configuration, the *FBM* creates and configures the *Adapter Instance*. The *Instance* then uses its *Engine* to fetch the data and display the respective HTML to the user. Thereafter, the user analyzes the data and selects the desired columns and other relevant features.

The full configuration received from the *RM* is then forwarded to the REST Adapter Instance. An example of the full configuration can be seen as following:

Mapper Input:

```
[forrest url="http://localhost:3001/merphelper/api/Order/all"]
        [field mapTo="name" typ="String"]name[/field]
        [field mapTo="projectManager" typ="String"]projectManager[/field]
        [convert]
                [LINKEDANALYZER id="20"]order[/LINKEDANALYZER]
        [/convert]
[/forrest]
```

Figure 40: Example of Full Configuration to LSH

As soon as the full configuration is received, it is converted to the desired JSON format. The REST adapter fetches the data from the REST endpoint and creates a JSON Array based on the *field* parameters defined by the user in the input configuration. The converted JSON in then forwarded to the corresponding Analyzer Adapter, together with the desired schema. The respective configuration is thereafter, used to create the java class, using the string builders. Once the class is created, the instances are created for each JSON object in the received data. The instances are pushed to the linked list, which after inserting the overall data, writes it to the file system, to be read and processed in the future. A small code snippet for inserting data in to a linked list can be seen as follows.

```
String classToRead = "ORDER";
Class c = Class.forName(classToRead);

for (int i = 0; i < ja.length(); i++) {
        jsonObj = ja.getJSONObject(i);
        Object obj = c.newInstance();
        Field[] fs = c.getDeclaredFields();
        for (Field field : fs)
        {
            field.setAccessible(true);
            Object fieldName = field.getName();
            Object fieldValue = jsonObj.get(field.getName());
            field.set(obj, fieldValue);
        }
        list.insert(new LinkedListNode<Object>(obj));
    }
String location = new File(".").getCanonicalPath() + "/src/main/resources/" + classToRead;
list.saveListToFile(location);
```

Figure 41: Creating Linked List and Saving to File System

Since, the functionality is based on the dynamic creation of the *classes*, therefore, there was a need to create a generic linked list, which could hold any specified class as its input. The generic code for the linked list used in the above code can be seen below.

```
public class GenericList<T>  implements Serializable {

    private LinkedListNode<T> first = null;
    public void insert(LinkedListNode<T> node) {
        node.setNext(first);
        first = node;
    }
    public void remove(){
        if(first.getNext()!=null)
            first = first.getNext();
        else first = null;
    }
    private void printList(LinkedListNode<T> node) throws IllegalArgumen-
tException, IllegalAccessException {
        for (Field field : node.getValue().getClass().getDeclaredFields()
) {
            field.setAccessible(true);
            String name = field.getName();
            Object value = field.get(node.getValue());
            System.out.println(name);
            System.out.println(value);
        }
        T value =  node.getValue();
        if(node.getNext()!=null) printList(node.getNext());
    }
    public void print() throws IllegalArgumentException, IllegalAccessEx-
ception{
        printList(first);
    }
    public void saveListToFile(String fileLocation) throws IOException{
        FileOutputStream fop=new FileOutputStream(fileLocation);
        ObjectOutputStream oos=new ObjectOutputStream(fop);
        oos.writeObject(this);
    }
}
```

Figure 42: Dynamic Linked List Class

Once the list is saved on the file system, it can later be retrieved and then processed based on the configurations provided by the user. One such processing option could be the sorting of the list. The data generated by processing the list is converted to JSON array and is sent back to the *RM*.

The data collection and conversion framework significantly reduces the user efforts by making the process of the data collection much more flexible. It allows the user to simply provide the configuration for the desired operation. Thereafter, the whole process is automated to collect and convert the data. Once the data is collected, cleaned and converted to the required format, the user has the option to run specified queries on the data to conclude useful results.

5 RESULTS & DISCUSSION

The purpose of this Chapter is to present the results of the examination and investigation that have been performed in perspective of the issue characterized in the first chapter of this thesis. The discoveries found over the span of the investigation alongside the constraints confronted, have been explained in this chapter. This portion talks about how the aforementioned mentioned issues have been tackled and highlights the results and analysis in the light of the implementation done in chapter 4.

The challenges faced due to the collection of data, its conversion and analysis, have been the base of this thesis. This thesis revealed the significance of data and how its quality could hinder the objectives of the organizations. The challenges posed by the collection of data, its cleaning and transformation have been thoroughly discussed. Data is an integral part for successful functioning of the production systems and therefore, cannot be overlooked. The collected data, presents a potential analysis for the system, which is a basic part for organizations to acknowledge the difficulties and issues experienced. The implemented framework was designed with these specific considerations in view. This section will highlight the findings and results of the approach that has been taken to address the pertaining problems.

1) Time required for cleansing of data

As mentioned previously in the review, Redman [27] argues that about 25 percent of the data collected is usually faulty. This issue has been dealt with carefully during the implementation of the framework. Any inaccuracies in the data caused due to the repetitive errors could easily be identified and be catered easily. Before the data can be collected, the user can provide a regex string, which is applied across all the data of the corresponding column. Therefore, filtering any undesired fields, which are not required by the user in the output.sss

2) Conversion of Data to desired format

Mapping data in to desired format firstly requires a well-formatted input source and secondly requires programming efforts for conversion. Thereafter, the issues of integration with the existing systems, takes additional efforts and time. This first issue is solved by selecting the desired fields and cleansing the data. However, the second issue requires only creating an adapter that takes JSON array as input and thereafter, converts the data in to the specified format. Since, all of the adapters are being created on the same platform; thereby it additionally solves the integration issues as well.

3) Budget issues for maintaining legacy systems

As discussed earlier, the cost of managing the legacy systems and their integration with the modern systems is a major barrier for the companies to upgrade their systems. The platform provided by the framework does not only solves the problem of integration, but additionally provides the merging of the system with the cloud as well. By converting the data in the desired format and passing it to the cloud, opens up the numerous opportunities and the ways this data could be put to use.

4) Managing new/unforeseen data formats

The Mapper provides a provides a simple yet powerful way for tackling the new data sources, which are required to be merged in to the already built platform. Other than configuring the new function block, the user needs to update the grammar for the newly introduced block. The grammar tells what the function block expects as the input, and thereafter parses the high level input to convert it in to the respective configuration. This provides user with the flexibility of hot deployment of the new data sources.

5) Tiresome process of collecting data, processing it and then analyzing it

Instead of using various tools for collecting the data, cleaning it and then converting it to the desired format, the implemented framework presents a unified platform for handling all of the mentioned processes. This saves the users and programmer's time from learning new tools and software in order to achieve the desired results. Therefore, by providing a central platform for multiple purposes, the framework reduces the cost as well as organizations time for managing, contemplating and analyzing the desired data sources.

6 CONCLUSION

This chapter sums up the research work done in this thesis and emphasizes on the achievements accomplished while implementing the proposed system. The challenges posed and the limitations faced while implementing the thesis will also be highlighted as well. Moreover, new suggestions and possible further developments are also argued in this section.

Production systems generate enormous amounts of data, while creating products and providing the respective services. It is essential to make a connection between both of these entities, since the production of goods is useless if it is not combined finely with the production of services. However, accessing data from multitude of sources for extracting, aggregation, converting and analyzing purposes remains an enormous challenge. Companies usually have legacy as well as advanced systems incorporated within their production facilities. However, modifying the systems either to acquire data from the legacy systems or to tackle new data formats often requires architectural changes, thus making the system more complicated. Therefore, construction of a loosely coupled system, where organizations could simply update or manage handling of diverge data sources, would bring much needed flexibility for the integration purposes. Therefore, by bridging the communication gap between the production of goods and those of the services would help the organizations to boost their profits.

The research of this thesis primarily focused on the serial integration of the function blocks methodology, by conducting the implementation on the REST, Excel, Linked List and SQL use cases. The main agenda was to provide a flexible system, which is generic enough to handle the new data sources, without the need to restart the system. The research is successfully accomplished on the Legacy System Hub, which is the core component of the C2NET project for collecting data from heterogeneous sources.

6.1 Accomplishments

The thesis has successfully achieved the objectives that were set during the initial phases of the thesis. The Legacy System Hub of the C2NET project was the major component of the research conducted in this thesis. The platform has succefully tackled the challenges posed by the diverge data sources, that needed to be merged with the existing systems. Thereby, providing much needed flexibility to run the production systems in a more cost effective and efficient way. The project has been able to provide a platform for the companies to boost their production chains. The Data Collection and Conversion framework of the C2NET project is effective enough to collect as well as convert the data from the heterogeneous legacy systems. Moreover, the extra effort required for creating new user interfaces, has been abstracted away with the help of the mapper. The user only

needs to provide the grammar for the new source, which is enough to configure the desired data endpoint. In addition, to all the provided functionality, the framework allows the user to run analyses on the converted data. The user is only required to provide a configuration defining the data source and the desired query.

6.2 Challenges and Limitations

Although, the implementation has successfully met the objectives of the thesis, however, some of the limitations were faced during implementation due to the constraints posed by the time. Firstly, the mapper is not generic enough and only provides the functionality for the text, REST and the excel adapters. In order, to provide support for the additional data sources, the mapper needs to be updated by providing the desired grammar for the new source. Moreover, only the functionality for the text, REST and excel adapters have been implemented on the Legacy System Hub (LSH). Data is collected only from the aforementioned sources; however, new data adapters could easily be created and integrated in to the platform. On the other hand, for the analysis and processing purposes, only the analyser adapters of SQL and linked list have been implemented. In addition to all this, the result of the analysis or the processing is only provided to the user in a JSON format.

6.3 Future Prospects

The research conducted in this thesis has opened numerous research questions in the respective field. Since, this research work only focused on the implementation of REST, text and excel adapters, new sources could be merged in to the platform. All the user is required is to program the adapter for the specific source and thereafter, providing an updated grammar for the configuration of the data source.

Currently, the user is required to manually change the lexer and parser grammars for the mapper, in order to successfully merge the new data source. The future research work could offer a layer of abstraction by providing a user interface, where they could configure the required grammars.

The current analyzer of adapters for the framework only include the SQL and linked list analyzers. These analyzer adapters accept the input data from the required data adapters and store them in the destined data endpoint. Additional analyzer adapter could be created according to the desired operations.

The current implementation returns the result of the analyzer in JSON format. However, according to the needs of the user, they could create an additional adapter, which could take JSON input and would convert the data in to desired format, e.g. XML.

Moreover, the data is sent directly to the user, without providing any graphical representation of the results being obtained. Future research work could merge a visualization tool with the current framework, which could display results as graphical or tabular format.

Additionally, the analysis of functionality of the framework could be extended by adding specific functionalities to certain Analyzer Adapters. In addition to performing simple queries, the useful information related to the data could also be discovered. For example, functions for calculating mean, standard deviation and frequency distribution can be added for the data under observation. Furthermore, in future work functionality can be provided for finding covariance and other relations between different attributes of the data, thereby, providing useful insights in to the respective data. This would help the user to draw useful inferences from the provided data. However, this type of analysis vary from case to case, but by combining the functionality with the extendable framework, the process could be made easier by using a common language, thus providing higher abstraction level for the user.

For future work, more functionality could be added to the analyzer adapter so that it can communicate to big data tools. For example in Apache Hadoop based architecture, Hadoop consumes data from other source/ multiple sources. Hadoop is used for processing big dataset using MapReduce programming model. Another example could be Apache Flume, it's a reliable, distributed and service to process large amount of data. Flume is used to ingest high-volume data to HDFS (Hadoop Distributed File System). This framework can be connected to Flume to provide structured data saved in DB.

This framework carries enormous potential to become a fascinated data collection, conversion and analysis environment. By providing the user with multitude of options, from data collection to analysis, the framework allows numerous functionalities to be added in to one amalgamated platform, thus providing a solution to integration issues in a cost effective manner.

REFERENCES

[1] Cormen, Thomas H. *Introduction to algorithms*. MIT press, 2009.

[2] Anany, Levitin. "Introduction to the design and analysis of algorithms." Villanova University (2003).

[3] Alsuwaiyel, Muhammad H. Algorithms: Design Techniques and Analysis (Revised Edition). Vol. 14. World Scientific, 2016.

[4] Sedgewick, Robert, and Philippe Flajolet. *An introduction to the analysis of algorithms*. Addison-Wesley, 2013.

[5] Karumanchi, Narasimha. Data Structures and Algorithms Made Easy in Java: Data Structure and Algorithmic Puzzles. CareerMonk Publications, 2013.

[6] Khan Academy. (2017). *Algorithms | Computer science | Computing |Khan Academy.* [online] Available at:https://www.khanacademy.org/computing/computer-science/algorithms#asymptotic-notation [Accessed on:27 Aug. 2017].

[7] Wirth, Niklaus. *Algorithms+ data structures= programs*. Prentice Hall PTR, 1978.

[8] Studytonight.com. (2017). Introduction to Data Structures | Data Structure Tutorial | Studytonight. [online] Available at: http://www.studytonight.com/data-structures/introduction-to-data-structures [Accessed on :27 Aug. 2017].

[9] Weiss, Mark Allen. *Data structures and algorithm analysis in Java*. Vol. 2. Addison-Wesley, 2007.

[10] Deitel, Paul J., and Harvey M. Deitel. *C++ how to program*. PearsonPrentice Hall, 2008.

[11] Koubarakis, M. (2009). An Introduction to Linked Data, (February), 1–14.

[12] Nehra, Ekta. "LINKED LIST IMPLEMENTATION USING C LANGUAGE: A REVIEW.", 678–687.

[13] Black, P. (2017). *linked list. Xlinux.nist.gov*. [online] Available at: https://xlinux.nist.gov/dads/HTML/linkedList.html [Accessed on:26 Aug. 2017].

[14] Barne, Granville, and Luca Del Tongo. "DSA." (2008).

[15] www.tutorialspoint.com. (2017). *Data Structures and Algorithms Tree*. [online] Available at: https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm [Accessed on: 27 Aug. 2017].

[16] Pushpa, Suri, and Prasad Vinod. "Binary search tree balancing methods: A critical study." *IJCSNS International Journal of Computer Science and Network Security* 7, no. 8 (2007): 237-243.

[17] Khese, C. S. (n.d.). Binary Search Tree and Its Applications : A Survey, 6200–6203.

[18] CLIFFORD, SHAFFER AA. "Practical Introduction to Data Structures and Algorithm Analysis." (2004).

[19] Garg, Deepak, and Megha Tyagi. "Comparative Analysis of Dynamic Graph Techniques and Data Structure." *International Journal of Computer Applications* 45, no. 5 (2012).

[20] IEC 61499-1/Ed.2: Function blocks - Part 1 (2012), Architecture. International Electro technical Commission, IEC,. [online] Available at: http://www.iec.ch/. [Accessed on: 27 Aug. 2017].

[21] Strasser, Thomas, Alois Zoitl, James H. Christensen, and Christoph Sünder. "Design and execution issues in IEC 61499 distributed automation and control systems." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41, no. 1 (2011): 41-51.

[22] Schmeling, Matthias. "Effective Visualization of IEC 61499 Function Blocks: With the CAKeFEED Function Blocks Editor." PhD diss., 2010.

[23] Jean Pierre Lorré. (2015). 1st Data collection from resources virtualization: legacy systems integration Work package [ONLINE] Available at: http://c2net-project.eu/documents/10184/232801/D3.3_636909_1st+Data+collection+from+resources+virtualisation.pdf/6093934d-be4f-4614-8094-1766ddff344d. [Accessed 20 August 2017]

[24] Virta, Jouko. "Application integration for production operations management using OPC Unified Architecture." MSc.(Tech.) thesis, Faculty of

Elec-Especificación y diseño de un sistema de gestión de campañas Batch, basado en el estándar internacional de automatización ISA-95 (2010).

[25] Williams, T. 1998. Enterprise integration in the process industries. Presented at theWorldBatch Forum 1998.

[26] Kitchin, Rob. The data revolution: Big data, open data, data infrastructures and their consequences. Sage, 2014.

[27] Redman, Thomas C. Data driven: profiting from your most important business asset. Harvard Business Press, 2008.

[28] G. K. Tayi and D. P. Ballou, “Examining data quality,” Communications of the ACM, vol. 41, no. 2, pp. 54–57, 1998.

[29] B. U. Libraries, “Boston University Data Services – Data Services at Boston University.” [Online]. Available: https://www.bu.edu/data/. [Accessed: 05-Dec-2017].

[30] Statistic Canada (2015). *Data Collection, Capture and Coding* [online] Available at: http://www.statcan.gc.ca/pub/12-539-x/2009001/collection-collecte-eng.htm [Accessed 20 August 2017]

[31] Haug, Anders, Jan Stentoft Arlbjørn, and Anne Pedersen. "A classification model of ERP system data quality." *Industrial Management & Data Systems* 109, no. 8 (2009): 1053-1068.

[32] Haug, Anders, and Jan Stentoft Arlbjørn. "Barriers to master data quality." *Journal of Enterprise Information Management* 24, no. 3 (2011): 288-303.

[33] Cappiello, Cinzia, Chiara Francalanci, and Barbara Pernici. "Time-related factors of data quality in multichannel information systems." *Journal of Management Information Systems* 20, no. 3 (2003): 71-92.

[34] Ballou, Donald, Stuart Madnick, and Richard Wang. "Special section: Assuring information quality." *Journal of Management Information Systems* 20, no. 3 (2003): 9-11.

[35] Feldman, Susan. *The high cost of not finding information*. Information Today, Incorporated, 2004.

[36] Statistic Canada (2015). *Data Collection, Capture and Coding* [online] Available at: http://www.statcan.gc.ca/pub/12-539-x/2009001/collection-collecte-eng.htm [Accessed 20 August 2017]

[37] Marshall, Catherine, and Gretchen B. Rossman. *Designing qualitative research*. Sage publications, 2014.

[38] Hitchcock, Graham, and David Hughes. Research and the teacher: A qualitative introduction to school-based research. Psychology Press, 1995.

[39] Shamoo, Adil E., and David B. Resnik. *Responsible conduct of research*. Oxford University Press, 2009.

[40] Schutt, Rachel, and Cathy O'Neil. *Doing data science: Straight talk from the frontline*. " O'Reilly Media, Inc.", 2013.

[41] Schwandt, Thomas A., and Thomas A. Schwandt. *Dictionary of qualitative inquiry*. Vol. 31. Thousand Oaks, CA: Sage Publications, 2001.

[42] Gottschalk, Louis A., and Fernando Lolas. "The Gottschalk-Gleser content analysis method of measuring the magnitude of psychological dimensions: Its application in transcultural research." *Transcultural Psychiatric Research Review* 26, no. 2 (1989): 83-111.

[43] Shephard, Roy J. "Ethics in exercise science research." *Sports Medicine* 32, no. 3 (2002): 169-183.

[44] Schneier, Bruce. Secrets and lies: digital security in a networked world. John Wiley & Sons, 2011.

[45] Microsoft - TechNet, Windows (2000). *Server, Library. Common Types of Network Attacks*. [Online]. Available at: http://technet.microsoft.com/en-us/library/cc959354.aspx [Accessed on: 28.09.2017].

[46] OWASP (2016). *The Open Web Application Security Project. Category: Vulnerability*. [Online]. Available at: https://www.owasp.org/index.php/Category:Vulnerability [Accessed on: 28.09.2017].

[47] Hypponen, MH (2011). *Defending the Net. TEDxBrussels*. [Online]. Available at: http://www.tedxbrussels.eu/2011/speakers/mikko_h_hypponen.html [Accessed on: 28.09.2017].

[48] Navy & Marine Corps WW11. *Commemorative Committee. Navajo Code Talkers: World War 11 Fact Sheet*. [online]. Available at: http://www.history.navy.mil/faqs/faq61-2.htm [Accessed on: 28.09.2017].

[49] Deng, J. Introduction to Symmetric Block Cipher. University of Colorado, lecturing material. [Online]. Available at: www.cs.colorado.edu/~jrblack/class/csci7000/f03/talks/7000_1.ppt [Accessed on: 28.09.2017].

[50] Bisbal, Jesús, Deirdre Lawless, Bing Wu, and Jane Grimson. "Legacy information systems: Issues and directions." *IEEE software* 16, no. 5 (1999): 103-111.

[51] Erlikh, Len. "Leveraging legacy system dollars for e-business." *IT professional* 2, no. 3 (2000): 17-23.

[52] Lin, Chang-Yang. "Migrating to relational systems: Problems, methods, and strategies." *Contemporary Management Research*4, no. 4 (2008).

[53] Weiderman, Nelson H., John K. Bergey, Dennis B. Smith, and Scott R. Tilley. *Approaches to Legacy System Evolution*. No. CMU/SEI-97-TR-014. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1997.

[54] Morello, D., A. Kyte, and B. Gomolski. "The quest for talent: You ain't seen nothing yet." *Gartner Inc. Retrieved Jul* 11 (2007): 2008.

[55] Paradauskas, Bronius, and Aurimas Laurikaitis. "Business knowledge extraction from legacy information systems." *Information technology and control* 35, no. 3 (2006).

[56] Aversano, L., & Tortorella, M. (2004). An assessment strategy for identifying legacy system evolution requirements in eBusiness context. Journal of Software Maintenance and Evolution: Research and Practice, 16(4-5), 255-276.

[57] Groover Jr, Mikell P. Automation, Production Systems and Computer-Aided Manufacturing. Prentice Hall PTR, 1980.

[58] Lean Manufacturing, “Production Management and Planning” [Online] Available at: http://www.lean-manufacturing-japan.com/interviews/production-management-planning-part1.html [Accessed on: 20th October 2017]

[59] Frohm, Jörgen. Levels of Automation in production systems. Gothenburg: Chalmers University of Technology, 2008.

[60] Qureshi, Khurshid Ali. "Data Collection and Information flow management frame-work for industrial systems." (2017).

[61] isagraf. 2017. *IEC61499_function_block_model*. [ONLINE] Available at: http://www.isagraf.com/get/IEC61499_function_block_model.pdf. [Accessed 22 October 2017].

[62] isagraf. 2017. *IEC61499_application_model*. [ONLINE] Available at: http://www.isagraf.com/get/july07pdfs/IEC61499_application_model.pdf. [Accessed 23 October 2017].

[63] bigocheatsheet. 2017. *Big-O Algorithm Complexity Cheat Sheet*. [ONLINE] Available at: http://bigocheatsheet.com/. [Accessed 23 October 2017].

[64] TUT. 2017. *PLANTCockpit Open Source*. [ONLINE] Available at: http://www.tut.fi/plantcockpit-os/. [Accessed 29 October 2017].

[65] Geetha, S. (2012). Possible Challenges of Developing Migration Projects. International Journal of Computers & Technology, 3(3), 463-465

[66] Zhang, W., Berre, A. J., Roman, D., &Huru, H. A. (2009, October). Migrating legacy applications to the service Cloud. In 14th Conference companion on Object Oriented Programming Systems Languages and Applications (OOPSLA 2009) (pp. 59-68).

[67] Khadka, R., Saeidi, A., Jansen, S., Hage, J., & Haas, G. P.(2013b). Migrating a Large Scale Legacy Application to SOA: Challenges and Lessons Learned. Proceedings of the 20th WCRE, Koblenz, Germany. IEEE

[68] smctraining. 2017. Automation Pyramid. [ONLINE] Available at: http://www.smctraining.com/en/webpage/indexpage/312. [Accessed 6 November 2017].

APPENDIX A – LEXER AND PARSER GRAMMAR

Lexer Grammar

```
lexer grammar MarkupLexer;
fragment C          : ('C'|'c') ;
fragment F          : ('F'|'f') ;
fragment I          : ('I'|'i') ;
fragment G          : ('G'|'g') ;
fragment R          : ('R'|'r') ;
fragment E          : ('E'|'e') ;
fragment D          : ('D'|'d') ;
fragment O          : ('O'|'o') ;
fragment W          : ('W'|'w') ;
fragment N          : ('N'|'n') ;
fragment L          : ('L'|'l') ;
fragment A          : ('A'|'a') ;
fragment S          : ('S'|'s') ;
fragment Y          : ('Y'|'y') ;
fragment H          : ('H'|'h') ;
fragment U          : ('U'|'u') ;
fragment T          : ('T'|'t') ;
fragment Q          : ('Q'|'q') ;
fragment P          : ('P'|'p') ;
fragment M          : ('M' | 'm') ;
fragment X          : ('X' | 'x') ;
fragment V          : ('V' | 'v') ;
fragment B          : ('B' | 'b') ;
fragment K          : ('K' | 'k') ;
fragment Z          : ('Z' | 'z') ;
fragment LETTERS    : [a-zA-Z] ;

OPEN                : '[' -> pushMode(BBCODE) ;
DOWNLOAD            : D O W N L O A D ;
CONFIGURE           : C O N F I G U R E;
ANALYSE             : A N A L Y S E;
SAYS                : S A Y S ;
SHOUTS              : S H O U T S ;
TEXT                : ~('[')+ ;
mode BBCODE;
CLOSE               : ']' -> popMode ;
SLASH               : '/' ;
EQUALS              : '=' ;
STRING              : '"' .*? '"' ;
SAMPLEID            : F I L E ;
FIELD               : F I E L D ;
FIELDARRAY          : F I E L D S;
RESTID              : R E S T ;
SQL                 : S Q L;
OUTPUTCONFIG        : O U T P U T C O N F I G U R A T I O N ;
URL                 : U R L;
HOSTNAME            : H O S T N A M E;
PATH                : P A T H;
```

```
USERNAME            : U S E R N A M E;
PASSWORD            : P A S S W O R D;
TYPE                : T Y P E;
NAME                : N A M E;
REGEXSTRING         : R E G E X S T R I N G;
DRIVER              : D R I V E R;
ADDRESS             : A D D R E S S;
SCHEMANAME          : S C H E M A N A M E;
TABLE               : T A B L E;
TABLENAME           : T A B L E N A M E;
FORFILE             : F O R F I L E;
FORREST             : F O R R E S T;
FILENAME            : F I L E N A M E;
RESTNAME            : R E S T N A M E;
PATHNAME            : P A T H N A M E;
IDNAME              : I D;
LINKEDANALYZER      : L I N K E D A N A L Y Z E R;
SQLANALYZER         : S Q L A N A L Y Z E R;
EXCELANALYZER       : E X C E L A N A L Y Z E R;
CLASS               : C L A S S;
CONVERT             : C O N V E R T;
SEPARATOR           : S E P A R A T O R;
ID                  : LETTERS+ ;
WS                  : [ \t\r\n] -> skip ;
```

Parser Grammar

```
parser grammar MarkupParser;

options { tokenVocab=MarkupLexer; }

line                    : (specsforfile | specsforrest | specsforex-
                  cel |             rest | excel | file | analyse |
                  congifure);
// Declare all the respective attributes needed ahead in the grammar
attribute               : ID '=' STRING ;
driverattribute         : DRIVER '=' STRING ;
addressattribute        : ADDRESS '=' STRING ;
schemaNameattribute     : SCHEMANAME '=' STRING ;
usernameattribute       : USERNAME '=' STRING ;
passwordattribute       : PASSWORD '=' STRING ;
tablenameattribute      : TABLENAME '=' STRING ;
filenameattribute       : FILENAME '=' STRING ;
restnameattribute       : RESTNAME '=' STRING ;
pathnameattribute       : PATHNAME '=' STRING ;
idattribute             : IDNAME '=' STRING ;
nameattribute           : NAME '=' STRING ;
separatorattribute      : SEPARATOR '=' STRING ;
typeattribute           : TYPE '=' STRING ;
```

```
url                 : '[' URL attribute* ']' content '[' '/' URL ']' ;
hostname            : '[' HOSTNAME attribute* ']' content '[' '/' HOST-
NAME ']' ;
path                : '[' PATH attribute* ']' content '[' '/' PATH ']'
;
username            :  '['  USERNAME  attribute*  ']'  content  '['  '/'
USERNAME ']' ;
password            : '[' PASSWORD attribute* ']' content '[' '/' PASS-
WORD ']' ;
type                : '[' TYPE attribute* ']' content '[' '/' TYPE ']'
;
name                : '[' NAME attribute* ']' content '[' '/' NAME ']'
;
regexString         : '[' REGEXSTRING attribute* ']' content '[' '/'
                    REGEXSTRING ']' ;
sampledownl         : DOWNLOAD ;
sampleconfi         : CONFIGURE ;
sampleanaly         : ANALYSE ;
content             : TEXT ;
element             : (content | tag) ;
tag                 : '[' ID attribute* ']' element* '[' '/' ID ']' ;

singlefield         : '[' FIELD attribute* ']' element* '[' '/' FIELD
']';

newsqlquery         : '[' SQLANALYZER idattribute driverattribute ad-
                    dressattrib ute schemaNameattribute usernameattrib-
                    ute passwordattribute ']' content '[' '/' SQLANA-
                    LYZER ']';
newlistquery        : '[' LINKEDANALYZER idattribute ']' content '[' '/'
                    LINKEDANALYZER ']';
wildentryquery      : '[' GENERICANALYZER typeattribute attribute* ']'
                    content '[' '/' GENERICANALYZER ']';

specsconvert        :  '['  CONVERT  ']'  (newlistquery*  newsqlquery*
                    wildentryquery*) '[' '/' CONVERT ']';
specsforfile        : '[' FORFILE nameattribute usernameattribute pass-
                    wordattribute  pathnameattribute separatorattribute
                    ']' singlefield* specsconvert '[' '/' FORFILE ']' ;
specsforrest        :  '['  FORREST  nameattribute  ']'  singlefield*
                    specsconvert '[' '/' FORREST ']' ;
specsforexcel       : '[' FOREXCEL nameattribute usernameattribute pass-
                    wordattribute  pathnameattribute separatorattribute
                    ']' singlefield* specsconvert '[' '/' FOREXCEL ']'
                    ;
excel               : sampledownl getexcel;
file                : sampledownl gettype element* ;
rest                : sampledownl getresttype element* ;
getexcel            : '[' SAMPLEID attribute* ']' name hostname path
          username  password type regexString '[' '/' SAMPLEID ']';
gettype             : '[' SAMPLEID attribute* ']' element* '[' '/' SAM-
                    PLEID ']';
```

```
getresttype        : '[' RESTID attribute* ']' element* '[' '/' RESTID
']';
multiplefields     : '[' FIELDARRAY attribute* ']' singlefield* '['
                   '/' FIELDARRAY ']';

sqlquery           : '[' SQL attribute* ']' element* '[' '/' SQL ']';
analyse            : sampleanaly sqlquery ;

command            : (DOWNLOAD | CONFIGURE | ANALYSE) ;

outconfig          : '[' OUTPUTCONFIG attribute* ']' multiplefields
                   element* '['  '/' OUTPUTCONFIG ']';
congifure          : sampleconfi outconfig ;
```